

Mutating Runtime Architectures as a Countermeasure Against Power Analysis Attacks

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

DISSERTATION

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)

von

Dipl.-Ing. Marc Stöttinger

geboren in Wiesbaden

Referenten der Arbeit:	Prof. Dr.-Ing. Sorin A. Huss Prof. Dr.-Ing. Georg Sigl
Tag der Einreichung:	19.09.2012
Tag der mündlichen Prüfung:	20.11.2012

Darmstadt, 2013
D17

Zusammenfassung

Diese Doktorarbeit beschäftigt sich mit der Erforschung von so genannten mutierenden Datenpfaden und ihrem Einsatz als Gegenmaßnahme gegen Leistungsanalyseangriffe auf die Implementierungen von kryptographischen Algorithmen. Mutierende Datenpfade sind konzeptionell Datenpfade einer Schaltung, welche während der Laufzeit ihre Architektur ändern können ohne die Korrektheit des implementierten Algorithmus zu kompromittieren. Das Konzept der mutierenden Datenpfade wird theoretisch in dieser Arbeit erforscht und anhand von Anwendungsbeispielen in der Praxis erprobt. Zur praktischen Umsetzung werden als Basisplattform FPGAs verwendet, weil die plattformspezifischen Eigenschaften das Kernkonzept der mutierenden Datenpfade geeignet unterstützen.

Leistungsanalyseangriffe gehören zur Klasse der passiven, nicht invasiven Implementierungsangriffe. Diese Art von Angriff nutzt die Leistungsaufnahme einer Implementierung zur Laufzeit aus, um geheime Parameter der kryptographischen Implementierung über ihr physikalisches Verhalten zu extrahieren. Bei diesen Angriffen wird das Gerät mit der Implementierung des kryptographischen Algorithmus im Normalbetrieb betrieben, so dass keine Spuren eines Angriffs nach Beendigung der Analyse erkennbar sind. Essentiell für diesen Angriff ist, dass der Widersacher weiss, welcher kryptographische Algorithmus im Gerät implementiert ist und dass er neben der Leistungsaufnahme auch Zugriff auf die Eingabe- und Ausgabewerte hat. Ebenso ist essenziell, dass sich das Gerät mit der Implementierung deterministisch verhält, so dass bei jeder Prozedur im Normalbetrieb die Schaltung die gleichen Operationen mit unterschiedlichen Eingabewerten ausführt. Genau dort setzt das Konzept der mutierenden Datenpfade an, um den Aufwand für einen solchen Implementierungsangriff unattraktiv für den Widersacher zu machen, da der zu betreibende Aufwand zum Extrahieren des parameterspezifischen Geheimnis der Schaltung zu groß oder gar unmöglich ist.

Im Gegensatz zu bisherigen Verschleierungs- oder Maskierungsmaßnahmen setzt das Konzept der mutierenden Datenpfade auf eine Verwürfelung der Schaltungsarchitektur des Datenpfades, um so die physikalischen Eigenschaften der Schaltung im Bezug auf die Leistungsaufnahme und die Ausführungszeit zu randomisieren. Es werden nicht wie beim Maskierungsverfahren die zu verarbeitenden Daten für die interne Bearbeitung randomisiert, sondern das physikalisch deterministische Verhalten der Schaltung manipuliert. Im Gegensatz zu manchen anderen Verschleierungsverfahren, welche auch das physikalische Verhalten der Schaltung verändern, wie bei *Dual-Rail-Logic*, ist die Änderung des physikalischen Verhaltens bei den mutierenden Datenpfaden nicht statisch sondern ändert sich kontinuierlich.

Konservativ betrachtet, werden verschiedene Verschleierungskonzepte, wie zum Beispiel Verwürfelung und Rauschgenerierung, genutzt um so einen mutierenden Datenpfad zu erstellen. Durch die geschickte Integration der verschiedenen Verfahren im Datenpfad können die verschiedenen Verschleierungsmethoden effizient als Gegenmaßnahme in eine Implementierung eines kryptographischen Algorithmus eingebettet werden. Zu diesem Zweck wird ein Entwurfsvorgang zur Erstellung von mutierenden Datenpfaden vorgeschlagen und diskutiert.

Summary

This thesis deals with the study of so-called mutating data paths and their utilization as a countermeasure against power analysis attacks on implementations of cryptographic algorithms. Mutating data paths are conceptually data paths of a circuit, which can change their architecture during runtime without compromising the correctness of the implemented algorithm. The concept of mutating data paths is investigated in this work theoretically and tested on the application examples for use in practice. A FPGA is used as basis platform for the practical implementation, because the platform-specific properties support the core concept of mutating data paths quite well.

Power analysis attacks belong to the class of passive, non-invasive implementation attacks. This type of attack uses the power consumption of an implementation during runtime to extract secret parameters of the cryptographic implementation by exploiting their physical behavior. For these kind of attacks, the device is operated with the implementation of the cryptographic algorithm in normal mode, so that no traces of the attack can be found after the analysis. Essential for this attack is that the adversary knows, which cryptographic algorithm is implemented on the device and that in addition he has access to the power consumption as well as access to the input and output values. It is also essential that the unit with the implementation behaves deterministically, so that the circuit performs the same operations with different input values for every execution in normal operation mode. Exactly at this spot the concept of mutating data paths tries to increase the costs for such an implementation attack in order to make it unattractive for the adversary. Thus, the additional effort to extract the specific parameters of the circuit, being too much or infeasible.

Unlike previous hiding based countermeasures or masking countermeasures the concept of the mutating data paths scramble the circuit of the architecture of the data path, so as to randomize the physical characteristics of the circuit in terms of power consumption and execution time. Instead of randomizing the data for the internal operations, as it is done in case of masking based procedures, the physical properties of the circuit are manipulated. The manipulation of the physical behavior in case of applying mutant data paths is not static but changes continuously, compared to other hiding techniques that also alter the physical behavior of the circuit, for instance *dual-rail logic*.

Conservatively, various hiding concepts, such as shuffling and noise generation, are used to create such a mutating data path. By the skillful integration of the various processes in the data path, the different hiding techniques are effectively embedded as a countermeasure in an implementation of a cryptographic algorithm. To this end, a design flow for the creation of mutant data paths is proposed and discussed.

Acknowledgment

Progressive research can not be done without a supportive and fruitful environment, as well as excessive and constructive discussions with open minded and excellent researchers. For sure, academic exchange influences the development and may be the sight of a researcher and thus, his work as well as this thesis. In this column I want to thank all the researchers, who allowed me to have such kind of positive experience during my research as an employee of the Integrated Circuits and System Lab.

I am very grateful to my supervisor Prof. Dr.-Ing. Sorin A. Huss, for his motivative and encouraging support and the joint constructive discussions during my PhD studies, as well for the opportunity to reach to the next level of academic education. I am very thankful to Prof. Dr.-Ing. Georg Sigl, for his interest in my work and for co-refereeing this thesis.

I like to thank my colleagues at both research departments, the Integrated Circuits and System Lab and the secure things in CASED, Tolga Arul, Tom Assmuth, Alexander Biedermann, Lijing Chen, Thomas Feller, Annelie Heuser, Adeel Israr, Attila Jaeger, Michael Kasper, Ralf Laue, Zheng Lu, Felix Madlener, H. Gregor Molter, Sunil Malipatlolla, Tim Sander, André Serin, Abdulhadi Shoufan, Hagen Stübing, Qizhi Tian, and Michael Zohner. I will miss the interesting discussions with them and the joint work on common challenges.

Research should not be restricted to a specific local area. Challenging as well as fascinating academic and industrial cooperations may grow from joint interest in a common research area. My thanks for having such inspiring cooperations and highlighting scientific discussions go to Ray Cheung, Thorsten Grawunder, Bernhard Jungk, Andreas Koch, Sascha Mühlbach, Steffen Reith, Oliver Stein, Thorsten Schütze, Werner Schindler and Gavin Yao.

I also want to thank Steven Arzt, Uwe Breidenbach, Oliver Dietz, Morteza Emamgholi, Octavio Gamero, Christopher Huth, Joannes Klaus, Dennis Kusidlo, Daniel Münch, Gregor Rynkowski, Christian Rückriegel, Jonas Schönichen, Alexander Spitzl and Christian Thaler, who supported my research by their conducted student projects, bachelor thesses, master and diploma theses.

My vast and grateful thanks go to my family, my brother, my sister in law, my parents, and my parents in law, who supported me on my academic journey over the years without any regard. Thank you to be always very insightful and patient to my elaborated academic working life and my perhaps sometimes excessive enthusiasm for my research. Thank you for the unquestioning support!

Notation

Variable types:

\mathbf{x}	: Vector
\mathbf{X}	: Matrix
X	: Random variable
x	: Realization of X
x	: Variable

Reserved identifier:

k	: A secret key of a cryptographic algorithm
k^c	: The correct (actually used) key of a crypto algorithm
$k_{(i)}$: The i-th subkey (a section) of the secret key k
$k_{(i)}^c$: The i-th correct (actually used) subkey of the secret key k
p	: Plaintext of a encryption or decryption scheme
c	: Ciphertext of an encryption or decryption scheme
$\text{GF}(p)$: Galois field over p
t	: Certain time instant
\mathcal{R}	: Noise
\mathcal{L}_t	: Side-channel leakage at t
$\tilde{\mathcal{L}}_t$: Estimated side-channel leakage (based on hypotheses) at t
$\mathcal{L}_{\mathcal{M}_t}$: Leakage model at t
HW	: Hamming wight model
HD	: Hamming distance model
\mathcal{F}_u	: Vector subspace of dimension u
$\mathcal{P}_{\mathcal{L}_t}$: Exploitable power consumption at time t
$\mathcal{P}_{\mathcal{OP}_{t,A}}$: Power consumption of operation A at t
$\mathcal{P}_{\mathcal{R}_t}$: Power consumption of the noise at t
σ	: Standard deviation
μ	: Mean value

Functions and operants:

$E(\mathbf{x})$: Expectation of \mathbf{x}
$\text{Var}(\mathbf{x})$: Variance of \mathbf{x}
$\text{Cov}(\mathbf{X})$: Covariance of \mathbf{X}
$\mathcal{L}_{\mathcal{F}_t}(\cdot)$: Leakage function
$\mathcal{D}(\cdot)$: Side-channel distinguisher
$a \oplus b$: a xor b
$a \wedge b$: a and b
$a \vee b$: a or b
$\neg a$: invert a

Contents

1	Motivation	1
1.1	Security Sensitive Application for Embedded Systems	1
1.2	Structure of the Thesis	3
2	Basics	5
2.1	Field Programmable Gate Arrays	5
2.1.1	Technology Platform	5
2.1.2	(Re-)Configuration of a FPGA	8
2.2	Side-Channel Attacks	10
2.2.1	Power Analysis Attacks	12
2.2.1.1	Exploitable Power Consumption of CMOS Circuits	15
2.2.1.2	Attack Methods	18
2.2.2	Countermeasures against Power Analysis Attacks	20
3	Concept of Mutating Runtime Architectures	23
3.1	Principles of Mutating Data Paths	24
3.2	Online Allocation:	
	Different Types of Processing Units	29
3.2.1	Influence on the Power Signature	30
3.2.2	Requirements for the Type of Process Unit Selection	33
3.2.3	Methods of Randomizing the Processing Units	36
3.3	Dynamical Binding:	
	Degree of Parallelism	42
3.3.1	Impact of Non-Static Parallel Processes on the Noise	43
3.3.2	Questions on how to Establish a Randomized Concurrency	47
3.3.3	Randomized Concurrency by Virtualization	51
3.4	Flexible Scheduling:	
	Progression of Tasks	56
3.4.1	Effect on Power Consumption Patterns	57
3.4.2	Partitioning the Algorithm	59
3.4.3	Randomized Micro-Program Selection	61
3.5	Design Flow of Mutating Data Paths	63
4	Analysis Framework	67
4.1	General Evaluation of Side-Channel	
	Resistance	68
4.1.1	Evaluation Procedure	72

4.1.2	Evaluation Metrics	75
4.2	CSA-Based Evaluation Metric	79
4.2.1	Constructive Side-Channel Analysis	80
4.2.2	Signal-To-Noise Based Evaluation Metric	86
4.3	Acquisition Setup	90
5	Application I: Symmetric-Key Algorithm AES	95
5.1	Considerations about the Architecture	95
5.1.1	Partitioning of the AES Modules	96
5.2	Implementation of the Mutating Data Path	98
5.2.1	Mutating SBoxes	100
5.2.1.1	Implementation of Different SBoxes	101
5.2.2	Concurrent SBoxes Management	104
5.2.3	Merging Round and Routines	107
5.2.3.1	Flexible Micro-Programs	107
5.2.4	Side-Channel Analysis of the Data Path	109
5.3	Discussion of the Evaluation Results	111
6	Application II: Public-Key Algorithm ECC	115
6.1	Hardware Architecture Considerations	116
6.1.1	Partitioning of the ECC Coprocessor	117
6.2	Realization of the Mutating Data Path	118
6.2.1	Mutating Finite Field Multipliers	121
6.2.1.1	Implementation of the Different Finite Field Multiplier Units	122
6.2.2	Management of Concurrent Multiplication Units	125
6.2.2.1	Virtualization Scheme	126
6.2.3	Various Projective Coordinates	128
6.2.4	Side-Channel Analysis of the Data Path	130
6.3	Discussion of the Evaluation Results	133
7	Conclusion	137
A	Basics to the Concept Chapter	141
A.1	Information Theoretic Background	141
A.2	CAD Algorithms	141
A.2.1	Left-Edge Algorithm	141
A.2.2	ASAP	142
A.2.3	ALAP	142

B Different Dimensional Subspaces	143
B.1 Higher Dimensional Subspaces of the AES Example Circuit	143
B.2 Basis Vectors of a Lookup Table Functions	144
Bibliography	145
Own Publications	159
List of supervised student thesis and student projects	163

List of Figures

2.1	Architecture of a FPGA	6
2.2	Design flow of Xilinx fPGAs	9
2.3	SCA work flow for a cipher encryption algorithm	14
2.4	Power Consumption of CMOS-inverter	16
2.5	Countermeasure methods	20
3.1	Overview of countermeasure techniques and related platfroms . . .	24
3.2	Modularization of the design exploration space	26
3.3	Layer model for modularization	27
3.4	Scatter plot of the power consumption of different design	32
3.5	Jeffery divergence as a metric for different designs	34
3.6	Total correlation as a metric for different designs	35
3.7	Block diagram of the hardened eMSK multiplier	39
3.8	Example application for utilizing the CFGLUT5 components . . .	41
3.9	Impact of the concurrency on the power consumption distribution	44
3.10	Visualisation of Algorithm 5 for a point doubling algorithm for ECC	49
3.11	Middleware concept for dynamic binding	53
3.12	Virtualizable ECC architecture [SBH10]	56
3.13	Subroutine rearrangement by routine rearrangement	59
3.14	Addressing scheme of the micro programs	63
3.15	Design flow to create a mutating data path and its generic archi- tecture	64
4.1	AES circuit example	75
4.2	CPA results of two different AES implementations	76
4.3	Comparison of the different evaluation Metrics	77
4.4	Template-based power consumption characterization of two im- plementations	79
4.5	Results of an CSA evaluation	83
4.6	Scheme of the effects captured by different leakage models	84
4.7	CSA results of basic Virtex 5 components	85
4.8	SNR-based CSA evaluation	88
4.9	Acquisition setup	91
4.10	Layer model of the advocated communication between the host and the implemented algorithm	92
5.1	SW/HW partitions of the AES Mutate architecture	96
5.2	Applied methods on the AES Mutate architecture	97

5.3	Leakage function of an exemplary AES data path	99
5.4	SBox Evaluation	103
5.5	Scheme of the mutating data path	105
5.6	Processing elements of the AES Mutate	106
5.7	Results of the SNR metric analysis of AES Comp and AES Mutate	110
5.8	Analysis of the correctly found subkeys over number of traces . .	111
6.1	SW/HW partitions of the ECC Mutate architecture	116
6.2	Applied methods on the ECC Mutate architecture	117
6.3	Core Multiplier Evaluation	124
6.4	Interconnection for the data transfer	125
6.5	Composition of the scalar multiplication based on micro-programs	129
6.6	Captured traces for the SPA evaluation	131

List of Tables

2.1	Power analysis attack methods	19
2.2	Countermeasures Methods	22
4.1	Comparison between success rate ratio and \widetilde{SNR} ratio	89
5.1	Resource consumption of the different SBox implementations . . .	102
5.2	VLIW structure of AES with mutating data path	108
5.3	SNR levels of the different subkeys	110
5.4	Resource consumption of various AES designs	112
6.1	Resource consumption of the different SBox implementations . . .	124
6.2	VLIW structure of the ECC-coprocessor	127
6.3	Supported point representations of the ECC Mutate coprocessor .	128
6.4	Resource consumption of various ECC designs	133
B.1	Construction of the $\mathcal{L}_{\mathcal{F}_{\text{toy-AES,CSA}^{12}}}(k, p)$	144

List of Algorithms

1	Multiplication with eMSK $z \leftarrow x \cdot y$ [MSH09]	38
2	Novel Masking Scheme [JSG ⁺ 12]	40
3	Shuffling between m different types of processing units PU^m . . .	44
4	Randomized parallel execution on n different types of processing units PU_n (concurrently)	46
5	Identifying processing units for random concurrency	48
6	Virtualization by middleware concept [SBH10]	54
7	Point doubling projective, $[2]P = (X_3 : Y_3 : Z_3)$ based on algo- rithm from [CF05]	61
8	Point doubling Jacobian, $[2]P = (X_3 : Y_3 : Z_3)$ based on algorithm from [CF05]	61
9	Common side-channel evaluation of implementation I of algorithm A	74
10	Determination of the activity A for attacking the Montgomery scalar multiplication on E	120
11	High-radix Montgomery multiplication from [MHAS08]	122
12	Left-edge algorithm from [Mic94]	141
13	<i>As Soon As Possible</i> scheduling from [Mic94]	142
14	<i>As Late As Possible</i> scheduling from [Mic94]	142

CHAPTER 1

Motivation

Contents

1.1	Security Sensitive Application for Embedded Systems .	1
1.2	Structure of the Thesis	3

1.1 Security Sensitive Application for Embedded Systems

In today's modern communication technology based society embedded electronic devices get more and more into our daily life in order to make it more enjoyable. The tasks of these embedded systems are various, starting from opening an automatic garage door, over streaming video content to an entertainment system, or collecting medical information of a patient's body, to enabling us to withdraw money at an ATM. Most of these tasks require to transfer data between different components within the device or even between different devices. Due to the task of the application the information contained in the transferred data between the different devices is sensitive and has to be handled with caution. So that not any device using the same medium of transportation or communication protocol can just listen to the sensitive data, and thus, an adversary may be able to retrieve the information in order to exploit them or abuse them for her or his own non honorable intentions.

Theoretically this problem is solvable due to the area of cryptography. By encapsulating the security sensitive application with an appropriate crypto system only the enabled parties can utilize the sensitive information or can properly and securely communicate with each other. Intently the challenge to implement such applications into embedded devices was to cope with the computational overhead in order to perform appropriate cryptographic operations on the data that has to be protected. For some application areas, for instance real-time video encryption or secure routing of network traffic, the density of data is rather high and the small inexpensive processing units of micro-controller chips are unable to process that amount of data in the given time period. According to the operation

environment of some applications the usage of a personal computer or a cluster of computers is not possible based on mobility aspects, the energy consumption, or the heat generation. A proper solution to achieve more computational power in small devices with a considerable power consumption is to design and implement an application specific integrated circuit (ASIC) for the required cryptographic operations.

The fabrication of ASICs is only affordable for a market with a high sales rate in order to compensate the production costs in a short time interval. Therefore, the produced chips have to have a certain life cycle on the market in which they provide the user with the designed security sensitive application with cryptographic operations. In some application areas like video entertainment the secured core applications, for instance video codec, are developing so fast, that a life cycle of a special ASIC design is too short to be reasonable. The same holds for communication devices in the military section, they also have a strong demand of being able to update their gear to new and improved algorithms. Therefore, ASIC implementations are not beneficial in these areas for hardware based algorithm acceleration, so field programmable gate array (FPGA) based systems are favored.

Since 1999 a new attacking method has to be considered for the design of security sensitive applications on embedded systems. These so-called *side-channel analysis* attacks are used to reduce the mathematical problem of breaking a cryptographic algorithm to a feasible effort, by investigating the implementation of the cryptographic algorithm and exploiting design flaws to extract valuable information of the algorithm's secret. Therefore the adversary who is attacking the implementation, observes the physical behavior of the implementation in the device under attack while it is active in normal operation mode. Thereby, she or he exploits for instance the data-dependent execution time, the data-dependent power consumption or the data-dependent electromagnetic radiation of the implementation under attack. Thus, the adversary needs physical access to the device under attack in order to capture the emitted physical behavior and the input and output of the cryptographic algorithm. The passive and non-invasive conditions to mount such an attack make side-channel attacks so dangerous, because there may be no trace of a conducted attack on the device. Especially power analysis attacks are one of the most popular class of side-channel attacks and so are heavily investigated in academia as well as industry.

In the following the focus is put onto the power analysis attacks among the side-channel attacks. In order to mount a power analysis attack the adversary only needs a mid- to high-class oscilloscope (depending on the device under attack) and a personal computer. The ability to execute such an attack does not directly depend on the architectural properties of the device under attack. Thus, micro-controller based systems, ASICs, as well as HW/SW-codesign based

implementations may be vulnerable to such an attack, according to their implementation.

In the last decades a lot of research has been done in order to find effective countermeasures against power analysis attacks. The proposed countermeasures can be classified into the following two different classes: masking and hiding. Both principles are either applied on the cell level or on the algorithmic level of the implementations. But the better the countermeasures get, the more effective attacks are developed, so that the head-to-head race proceeds. Even so specially secured FPGAs for military applications are not perfectly secure against side-channel attacks, cf. [SW12]. Thus, several countermeasures are applied in a secured design to increase the effort of exploiting the data-dependent power consumption. Therefore, embedded devices with security sensitive applications are hardened against attacks on different levels preventing the leakage of exploitable information at high costs.

Unfortunately, as mentioned before, there are nearly no countermeasures against power analysis attacks on the architectural level between the algorithmic based countermeasure in the software and countermeasures on the cell level in the hardware. This rises the question if it is possible to integrate a suitable countermeasure concept on the architectural level of hardware/software codesign based implementations. Therefore, this thesis investigates the integration of several hiding based countermeasures on the architecture of implemented cryptographic algorithms. Furthermore a design flow for the so-called mutating data paths is introduced and discussed. The core idea behind the principle of mutating data paths is an architecture that varies during runtime, so that the functionality of the algorithm is maintained, while the physical behavior of the implementation changes non-deterministically.

1.2 Structure of the Thesis

The following second chapter provides the reader with an introduction to two basic areas used in this thesis to investigate the concept of mutating runtime architectures as a countermeasure against power analysis attacks. At first the structure of the used field programmable gate array based devices is introduced as well as its vendor specific design. The second section of Chapter 2 provides the reader with an introduction into side-channel analysis with the focus on power analysis. It is introduced how the data-dependent power consumption may be exploited to reveal secret parameters of a cryptographic implementation. Also, different power analysis attacks and different countermeasure techniques against such attacks are discussed shortly.

Chapter 3 describes the key concept that is investigated in this thesis. The concept of mutating data paths is described and discussed in theory. The concept of the mutating data path countermeasures is compared with the existing concepts of countermeasures and their application on the circuit. First a general overview over the fundamental principle on the mode of action of mutating data paths with respect to countermeasures against power attacks is presented. In this context the essential properties of a data path structure is discussed to be randomized during runtime in order to hinder an adversary to extract any exploitable power consumption from the implementation. Three different mutating methods are introduced section-wise to design a mutating data path in general. The methods are discussed in terms of the design specific properties, the effect on the exploitable power consumption signature, and their application, which is illustrated on small examples. At the end of the third chapter the design flow of generating a mutating data path is described based on the previously introduced mutating methods.

Chapter 4 is the core chapter about the side-channel analysis methods used to evaluate the mutating data path countermeasure. The theoretic concept of exploitable side-channel leakage is discussed in the first section of this chapter as well as the modeling of hypothesis to exploit the power consumption of a design via side-channel analysis attacks. The concept of constructive side-channel analysis is used as a fundament for the side-channel evaluation, therefore the concept of constructive side-channel analysis is introduced in this chapter as well. At the end of Chapter 4 a new signal-to-noise metric, based on the intermediate results of the methods of the constructive side-channel analysis is introduced, which allows a fair comparison of different implementations.

Chapter 5 and Chapter 6 provide the application on the mutating data paths concept on two different cryptographic algorithm schemes. In the fifth chapter the application of the mutating data path concept on a symmetrical-key algorithm is investigated. First the design properties of the design are discussed, followed by the implementation of the in Chapter 3 introduced mutating methods on the selected cipher algorithm. The implementation of the cipher, in this case an AES, is then configured on a special evaluation and a side-channel analysis on the design is mounted. With the same procedure Chapter 6 investigates the application of the mutating data path countermeasure on an elliptic curve cryptography coprocessor in order to discuss the countermeasure application on asymmetrical cryptographic algorithms.

A conclusion of the results and the proposed countermeasure investigated in this thesis is provided in the last Chapter 7. After a short summary, open questions for further research and improvements as well as applications on other designs are discussed.

Contents

2.1	Field Programmable Gate Arrays	5
2.1.1	Technology Platform	5
2.1.2	(Re-)Configuration of a FPGA	8
2.2	Side-Channel Attacks	10
2.2.1	Power Analysis Attacks	12
2.2.2	Countermeasures against Power Analysis Attacks	20

2.1 Field Programmable Gate Arrays

The focus of the thesis is on mutating runtime architectures as countermeasures against power analysis attacks on *Field Programmable Gate Arrays* (FPGA) platforms. Thus, this section provides some background information about the used technology platform as an introduction to the subject. First the internal structure and the functionality of a FPGA based silicon device is discussed followed by an introduction of the devices special functionality, the partial reconfiguration during runtime.

2.1.1 Technology Platform

FPGAs are filling the gap between the *Central Processing Unit* (CPU) of personal computers or a micro controller and so-called *Application Specific Integrated Circuit* (ASIC) in terms of flexibility, power consumption, and performance. While a CPU is very flexible by offering the ability to program an executable application via software, the application of an ASIC is fully customized and thus it only provides the implemented application specific functionalities. Nevertheless, ASIC provides a better power consumption and performances based on their application-specific optimization and therefore, specialized and fixed architecture. For instance, the data path of an AES-128 ASIC can be implemented fully parallel with a bit-width of 128 bits with a throughput of 128 bits per clock cycle.

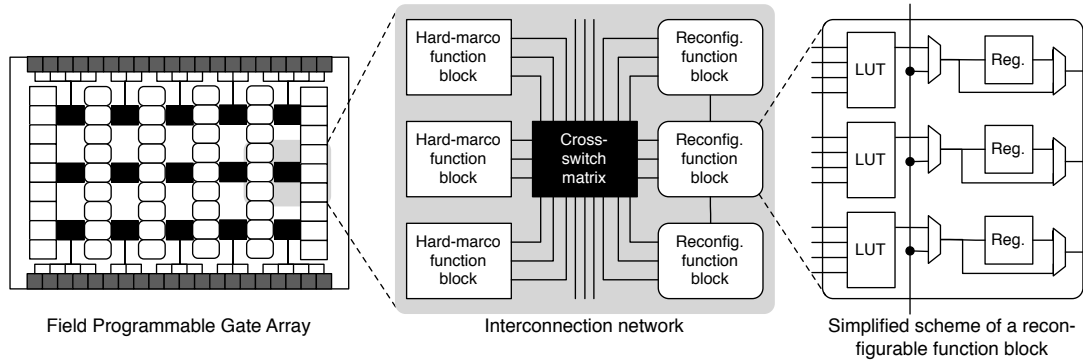


Figure 2.1: Architecture of a FPGA

The architecture of a FPGA provides a compromise between the flexibility of a CPU and the performance of an ASIC by utilizing a huge number of reconfigurable function blocks, hard-macro function blocks, and programmable interconnection networks to implement any arbitrary circuits. Cross-switch matrices and grid-structures of wires are exploited as routing elements to establish the interconnection between the basic components of the FPGA. The reconfigurable function blocks are used to realize a circuit providing the functionality of the implemented application. *Look-up table* (LUT) structures and registers are used as basic elements of these reconfigurable function blocks, c.f. Fig. 2.1. The technology architecture of such LUTs can either be small programmable memory-cells, multiplexer structures or programmable anti-fuses. Nowadays, the most common technology for LUTs in FPGAs are memory-based functions blocks, because they are multiple times programmable compared to the anti fuse-based LUTs. The memory-cell based LUTs can either be implemented using the *Complementary Metal Oxide Semiconductor* (CMOS) technology or using the *FLASH* technology, while the storage component¹ of the function block is usually in COMS technology [FPG].

In the following focus will be on the architecture of a Virtex 5 platform of the vendor Xilinx Inc. , which was used in order to conduct the experiments stated in this thesis. Please note that FPGAs from other vendors have slight differences in their internal structure and their functionality. The reconfigurable function blocks on the Virtex 5 platform are called *Configurable Logic Block* (CLB). Beside the CLB primitives other specialized hard-macro function blocks are embedded in the Virtex 5 architecture. These customized components are also attached to the interconnection network in order to combine their functionality with the CLBs to implement a specific application on the FPGA. The essential basic primitives

¹register or transparent latches

of the Virtex 5 FPGA platform, used to implement an arbitrary circuit are listed as seen below:

CLB The Configurable Basic Block of the Virtex 5 architecture consists of two so-called slices. A slice is the main resource to establish the Boolean functions for any arbitrary sequential and combinatorial circuit. Each slice is regularly constructed out of four six-input LUT, interconnection logic, and four storage elements, which can be configured as register or transparent latch. The six-input LUT is a *Static Random Access Memory* (SRAM) based configurable lookup-table and it is (re-)configurable before or during runtime to express a Boolean function with up to six parameters. Actually, the six-input LUT is assembled by two five-input LUTs and a multiplexer, which switch the output of the two LUTs according to the sixth-input bit to the main output of the LUT. The second five-input LUT is additionally connected to an auxiliary output of the six-input LUT primitive, which enables a designer to implement two Boolean functions with five parameters into one of six-input LUT components.

BRAM The so-called *BRAM* component is a block primitive of 36Kb *Random Access Memory* (RAM). This storage component is structured as a RAM component with maximum 36-bit width for data and an address depth of 1024. The memory content of this primitive is accessible via two independent synchronous or asynchronous configurable ports. With those properties of a true dual-port memory block the BRAM can be exploited as a *First-In, First-Out* (FIFO) component in order to connect two circuits working in different clock domains. The initialization option of this component enables one to use this BRAM as a *Read-Only Memory* (ROM) component in order to provide pre-calculated intermediate values or realize instruction memory for a controlling unit.

DSP48E A specialized arithmetic unit for integer value processing is integrated in the *DSP48E* hard-macro on the Virtex 5 platform. This unit offers 48-bit width logic and an adder/subtraction unit and two complement supporting multiplier units with two differently wide operands. The first operand is 18-bit width, while the second operand is 25-bit width. So this unit can be used to perform multiplications of two 18-bit width integer values. Due to the internal structure of this primitive it is possible to perform on the one hand typical digital signal processing operations like a *Multiplier-Accumulator* (MAC) operations and on the other hand 12-bit width *Single Instruction Multiple Data* (SIMD) linear arithmetic and logic operations. Furthermore, this unit offers up to 3 pipeline states in the unit to improve the maximal throughput.

2.1.2 (Re-)Configuration of a FPGA

An arbitrary circuit can be implemented on the Virtex 5 platform by configuring the FPGA with a so-called bitstream after the power up phase. A bitstream consists of header information and configuration data to map the implemented digital circuit behavior on the FPGA platform. This configuration file is the final file format resulting in the end of the workflow of the implementation process. The regular work flow of the design process of a circuit for an Virtex 5 FPGA is portioned into several steps:

The regular design flow for a circuit design on FPGAs starts with the description of the circuit in a hardware description language (HDL), for instance VHDL. In the *Synthesis* process step is the circuit description translated into a net-list. This net-list consists of the basic components, which are available on the FPGA platform including the components settings. The next process step *Map* maps the net-list to the actual used FPGA type by allocating the needed physical basic components on the FPGA to implement the circuits based on the results of the synthesis step. Based on allocated components the physical implementation is done in the step *Place and Route* (PAR). First the allocated components are individually matched to available and fitting physical instances of the basic components. Then, the routing resources are utilized in order to connect the configured physical instances. After the PAR process step the bitstream is assembled based on the configuration settings of each used component of the FPGA platform. The bitstream can then be uploaded to the FPGA using the *Joint Test Action Group* (JTAG) interface or the Xilinx Inc. specific *Select Map* interface. It is also possible to store the bitstream on a so-called *platform flash*, which automatically configures the FPGA after the power up phase.

Figure 3.15a) depicts the regular design flow of a static circuit design for a Virtex 5 FPGA. With such a design flow the FPGA is configured with a customized circuit for a specific task, which does not need to change its functionality during the runtime of the FPGA. In case an adaptation of the functionality of the FPGA in terms of a circuit adjustment is required the FPGA design has to be designed partial reconfigurable. A reason for that request may be a low amount of resources that can be shared between a sequential order of dependent and isolated tasks. Thus, the so-called partial reconfigurable designs share basic components of the FPGA in a defined area in order to reconfigure their functionality according to the executed tasks. The Virtex 5 platform supports this feature due to its architectural nature of utilizing SRAM-Cells to store the configuration bits for the routing paths, the hard-macro configuration, and the Boolean function realization of the LUTs. In order to communicate between the different sequential configured circuits at least one static component is required, which organizes the data flow and manages the configuration of the partial re-

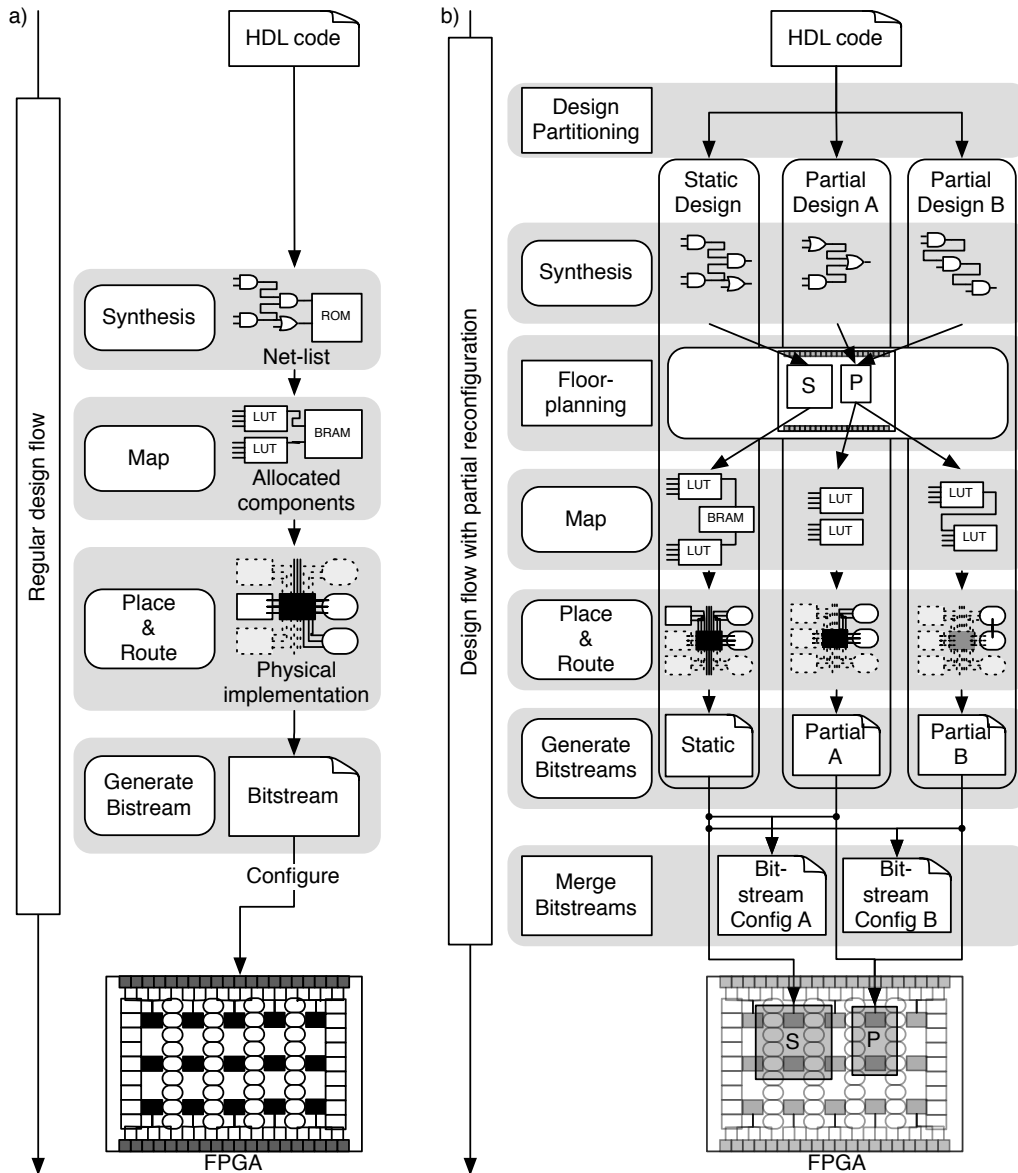


Figure 2.2: Design flow of Xilinx fPGAs

configurable area.

Generally speaking, the regular design flow is slightly extended with three additional processes to create a partially reconfigurable design. After the circuit design has been described in HDL the entire circuit has to be portioned into static and partial parts. One constraint of the in here introduced design flow is that the top-level circuit is always a static design with a black-box placeholder for the reconfigurable partial circuits. The different circuit designs for the shared resources of the FPGA area of the partial reconfiguration area have to be

assigned. For each partial reconfigurable area at least one circuit has to exist, even if the same circuit should be copied to different partial reconfigurable areas. After that each circuit design is individually synthesized to a net-list, so that for each fragment of the partition HDL-code a net-list exists. The next process step is to address the different areas of the FPGA and assign them to static or partial reconfigurable sections. During this *Floor planning* process step it is crucial to bind the specific net-lists to the according static or reconfigurable sections of the FPGA. Bounded to the constrained areas the physical implementation of each circuit as net-list is done individually tailored to the restrictions of the addressed section. After the MAP and PAR process steps the bitstream of each circuit, which is either assigned to a partial section or a static section, is generated. After this step the bitstreams are merged so that the bitstream of the top-level static circuit is merged with each of the reconfigurable circuit bitstreams in an own complete bitstream. The number of merged bitstreams depends on the number of partial reconfigurable sections on the FPGA and on the number of different reconfigurable circuits per partial reconfigurable section. Figure. 3.15b) depicts exemplarily the process steps of the design flow for a design with one static configuration circuit and one partial reconfigurable section on the FPGA with two different circuit designs for this section.

One advantage of the partial reconfigurable sections is that their multiple configurable circuits can be altered during runtime. The sections apart from the reconfiguring section can still be used during the reconfiguration process without losing their functionality. The so-called *internal configuration access port* (ICAP) manages the partial configuration of the reconfigurable sections of a FPGA from Xilinx Inc.. In case of a Virtex 5 platform each frame (the smallest addressable reconfigurable section on this platform) is controlled by this interface. This smallest addressable partial area includes hard-macro cells, CLBs and routing resources. This interface is fed with the partial bitstreams from a controller in order to reconfigure the partial reconfigurable sections. The controller can either be a micro blaze with a special software subroutine or a Final State Machine. Usually the partial bitstream is then stored in a BRAM on the FPGA, which is then utilized by the controller and the ICAP.

2.2 Side-Channel Attacks

In the late nineties the side-channel analysis attacks were discovered in the civil research community as a class of non-invasive implementation attacks. It is not sure since when the phenomena of side-channel analysis like attacks were studied in the military area. Implementation attacks target the implementation flaws of a cryptographic algorithm in order to extract the algorithm secret, for instance

the secret key of a cipher encryption algorithm. The tremendous threat of a non-invasive side-channel analysis (SCA) attack is the untraceable indication of a conducted attack. The attack is performed when the implementation is running in normal operation mode without any violation to its operational specifications, cf. [MPO07]

Physical Interaction Every device executing an algorithm, cryptographic or not, interacts with its physical environment during runtime. Based on the internal procedures of the implemented algorithm the activity of the device is controlled and thus the energy consumption and the runtime. Therefore, each algorithm has its own energy and processing time signature. Even more the processed data of the algorithm will lead to a specific signature of energy consumption or a specific execution time, if the processed data affects the activity of the device. In straightforward implementations this is the case due to data-dependent control flow changes like if-else-instructions or different amounts of switching activities based on internal computations.

By monitoring the activity of the device the adversary may extract information of intermediate values of internal operations or of the current internal state of the algorithm. The monitoring process of the device's activity is done by measuring one or more of its physical properties during runtime. The unintended information leakage from the measured physical properties are then exploited in order to verify hypotheses about the algorithms internal cryptographic secret. The hypotheses are expected values of the observed physical quantity caused by the adversary's suggested intermediate value changes or state transitions. Only very little knowledge about the actual implementation of the algorithm is needed by the adversary in order to construct these hypotheses. Based on the divide and conquer principle the hypotheses are formulated for every value in the search space for only a few bits instead for the complete bit width of the operation.

Exploitable Design Properties The first SCA attack was published in 1996 by Kocher [Koc96]. In this publication it is shown on the example of the RSA algorithm [RSA78] how the so-called timing attacks exploit data-dependent runtime variations of event-based control flow branches to gain knowledge of the secret key. Furthermore, Kocher demonstrated that with additional information based on simple time measurements public-key schemes like Diffie-Hellman key exchange [DH76] can be compromised. Three years later he and his coauthors introduced power analysis attacks in [KJJ99]. These side-channel attacks exploit the power consumption of a device running a cryptographic algorithm in normal operational mode to reveal information about the used secret key. Besides using the power consumption as measurable physical quantity other implementation behavior symptoms can be exploited to monitor the device's internal activity. In

2001 the so-called electromagnetic (EM) attacks were introduced, which exploit the inductive physical law to measure the data-dependent energy consumption of an implemented cryptographic algorithm, cf. [GMO01, QS01]. Both of the early works in this area [GMO01, QS01] demonstrate the practicability of these attacks by successfully extracting the secret key of the *Data Encryption Standard* (DES) [DES77] implementation. Another class of side-channel attacks are the so-called cache attacks [Pag02], which exploit the implementation of cryptographic algorithms on the architectural level and not directly on the gate-level.

2.2.1 Power Analysis Attacks

The focused side-channel attack methods in this thesis, the power analysis attacks, are one of the most established attack methods in the class of side-channel attacks and well known in the cryptographic community. Power analysis based side-channel attacks exploit the data-dependent power consumption, which occurs performing in internal operation at a certain point in time. By pattern matching over a vector of power consumption values \mathbf{p} the adversary extracts information or verifies hypotheses over internal secrets. The elements of \mathbf{p} can either be power consumption values of an operation \mathcal{P}_{OP} in a time series over the period of one measurement:

$$\mathbf{p}_t^T = (\mathcal{P}_{OP_1, t_0}, \mathcal{P}_{OP_1, t_1}, \dots, \mathcal{P}_{OP_1, t_m}) \quad (2.1)$$

or the elements can be power consumption values of an operation \mathcal{P}_{OP} at a certain point in time over a series of n measurements:

$$\mathbf{p}_s^T = (\mathcal{P}_{OP_1, t_0}, \mathcal{P}_{OP_2, t_0}, \dots, \mathcal{P}_{OP_n, t_0}). \quad (2.2)$$

For reasons of better distinction a \mathbf{p} vector containing elements from a time series from one measurement is further labeled by \mathbf{p}_t while the \mathbf{p} vector containing elements from multiple measurements at the same time instant is labeled by \mathbf{p}_s . In case the data-dependent power consumption is the focused information leakage, the power consumption values of a unique sequence of internal calculations are exploited. This sequence is constructed by performing the same operation with different input parameters several times, so that for each execution the internal calculation is performed with different intermediate values at the same certain point in time.

It is obvious that each performed measurement run has to be performed under the same conditions except for the varying input parameters, in order to assure that the varying power consumption values over the set of measurement

runs stems from the same operation. If this is not the case the vector \mathbf{p}_s contains randomized power consumption values from various operations and therefore the data-dependent, unique pattern of the internal operation's power consumption values is not exploitable. The following shows in detail how to extract these pattern from the acquired power traces and how to set up hypothesis and estimated power consumptions supporting the information extraction from the traces.

Information within the Power Traces The acquired power consumption data of n measurements with m captured samples for each measurement is represented in the $n \times m$ matrix \mathbf{P}_{Tr} :

$$\mathbf{P}_{Tr} = \begin{pmatrix} \mathcal{P}_{OP_1,t_0} & \mathcal{P}_{OP_1,t_1} & \cdots & \mathcal{P}_{OP_1,t_m} \\ \mathcal{P}_{OP_2,t_0} & \mathcal{P}_{OP_2,t_1} & \cdots & \mathcal{P}_{OP_2,t_m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}_{OP_n,t_0} & \mathcal{P}_{OP_n,t_1} & \cdots & \mathcal{P}_{OP_n,t_m} \end{pmatrix}. \quad (2.3)$$

The rows of the matrix \mathbf{P}_{Tr} are equal to the content of the \mathbf{p}_t -vector of one measurement over a certain time period enveloping the targeting cryptographic operation. The number of entries in the rows (number of sampled time instants n) depends on the capturing length of the recorded power consumption as well as on the sampling rate of the continuous power consumption signal. The column of the matrix \mathbf{P}_{Tr} contains the same information as the \mathbf{p}_s -vector. The number of column entries of \mathbf{P}_{Tr} depends of the number of measurements in the conducted measurement set. If all measurements in the conducted set of measurements are aligned the adversary can observe the influence of input parameter changes to the power consumption of the implemented cryptographic algorithm. On the one hand strong changes in the column of \mathbf{P}_{Tr} indicate different data-dependent power consumption between the time instants. On the other hand differences in the rows of \mathbf{P}_{Tr} indicate changes in the control flow of the algorithm depending on the variation of the input data.

Estimated Power Consumptions Due to the fact that the complete power consumption of the device is represented in \mathbf{P}_{Tr} monitoring, the specific activity of a certain part of the circuit is often not directly visible. Thus, hypotheses about the power consumption of a specific internal operation are formulated as stated before in Subsect. 2.2. For each value of the secret parameter of the targeting internal operation a hypothesis is set up, so that the number of hypothesis equals the number of elements in the search space. Each hypothesis provides an estimated power consumption according to the used input parameter at the conducted measurement run. The captured values in \mathbf{P}_{Tr} are then compared to every estimated power consumption derived from a hypothesis. The comparison

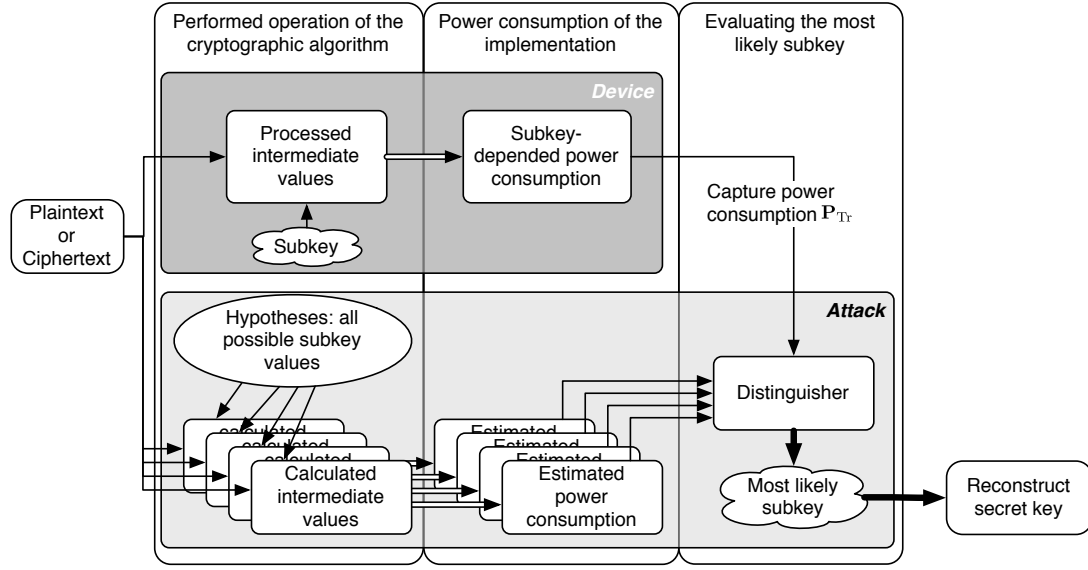


Figure 2.3: SCA work flow for a cipher encryption algorithm

is done by a so-called distinguisher that evaluates the most likely hypothetical power consumption and thereby references to the most likely correct hypothesis assumption. Usually, the distinguisher is a pattern-matching based, statistical or stochastic based evaluation algorithm.

Figure 2.3 depicts the above-described generic workflow of a power analysis attack on the example of a cipher encryption algorithm. In this case the target of the attack is the secret key, which is used for encryption or decryption in the cipher scheme, but the power consumption is exploited to get the subkeys (bit parts of the key). The obtained subkeys are then used in order to reconstruct the complete key following the divide and conquer principle. The device, in which the encryption algorithm is implemented, executes one encryption or decryption, depending on the operational mode, with a given plaintext or ciphertext for each power measurement. For each possible subkey value and for each plaintext or ciphertext a hypothesis is formulated and the corresponding estimated power consumption for each measurement run. The distinguisher compares the estimated power consumption values with the captured power consumption to determine the most likely correct subkey candidate. In case the intermediate values of an internal operation are the focus of the exploitation, the estimated power consumption values of each hypothesis are compared with the rows of \mathbf{P}_{Tr} . Further notes on processing a side-channel attack is provided more formalized in Sect. 4.1 in conjunction with leakage model construction.

2.2.1.1 Exploitable Power Consumption of CMOS Circuits

In this subsection the power consumption of CMOS-technology gates is discussed in order to characterize the data-dependent part of the exploitable power consumption during a side-channel attack. From a designing point of view CMOS technology has the advantage of having nearly no static power consumption, cf. [JB07]. Thus, the main power consumption bases on the activity of the implemented circuit. Nevertheless, the total power consumption $\mathcal{P}_{t,\text{tot}}$ at the time instant t of a CMOS circuit consist of three additive components:

$$\mathcal{P}_{t,\text{tot}} = \mathcal{P}_{\text{dyn}} + \mathcal{P}_{\text{sc}} + \mathcal{P}_{\text{stat}}. \quad (2.4)$$

The dynamic power dissipation in Eq. (2.4) is labeled with \mathcal{P}_{dyn} , the power dissipation caused by short circuits is denoted by \mathcal{P}_{sc} , and $\mathcal{P}_{\text{stat}}$ denotes the static power dissipation. Please keep in mind that Eq. (2.4) models the total power consumption ideally. In case of measuring $\mathcal{P}_{t,\text{tot}}$ the measurement noise $\mathcal{P}_{\mathcal{R}_t}$ has to be added for each point in time t :

$$\mathcal{P}_{\text{measure}} = \mathcal{P}_{\text{tot}} + \mathcal{P}_{\mathcal{R}}. \quad (2.5)$$

In the following the components of \mathcal{P}_{tot} and their impact on the exploitable power consumption will be shown in greater detail. For reasons of clarity the components will be elaborated on the example of a simple inverter CMOS-gate depicted in Fig. 2.4.

Static Power Dissipation The static power consumption in CMOS based technology bases on the leaking current and the supply voltage:

$$\mathcal{P}_{\text{stat}} = I_{\text{leak}} \cdot V_{\text{DD}}. \quad (2.6)$$

The leakage power dissipation occurs only if the corresponding MOS transistor is "switched off" and has a huge ohmic resistance between the source and the drain. This leakage current is a physical property of the CMOS technology and increases exponentially with the temperature. Three components define the leakage current in Eq. (2.6): the gate leakage current I_{gl} , the subthreshold leakage current I_{sl} and the junction leakage I_{jl} , cf. Fig. 2.4(a). The first component is the leakage current flow through the gate of the transistor to the drain. The gate leakage depends on the gate oxide thickness: the thinner it gets the more electrons are tunneling the gate oxide surface. The problem is usually addressed by using several high-k gate dielectrics to decrease the leakage current, cf. [RBNB04]. The second leakage component, the subthreshold leakage current, is the current flow from the source to the drain while the MOS transistor is in cut-off mode. It directly depends on the physical property of the threshold voltage, and thus increases for newer downscaled CMOS technologies. The junction leakage current

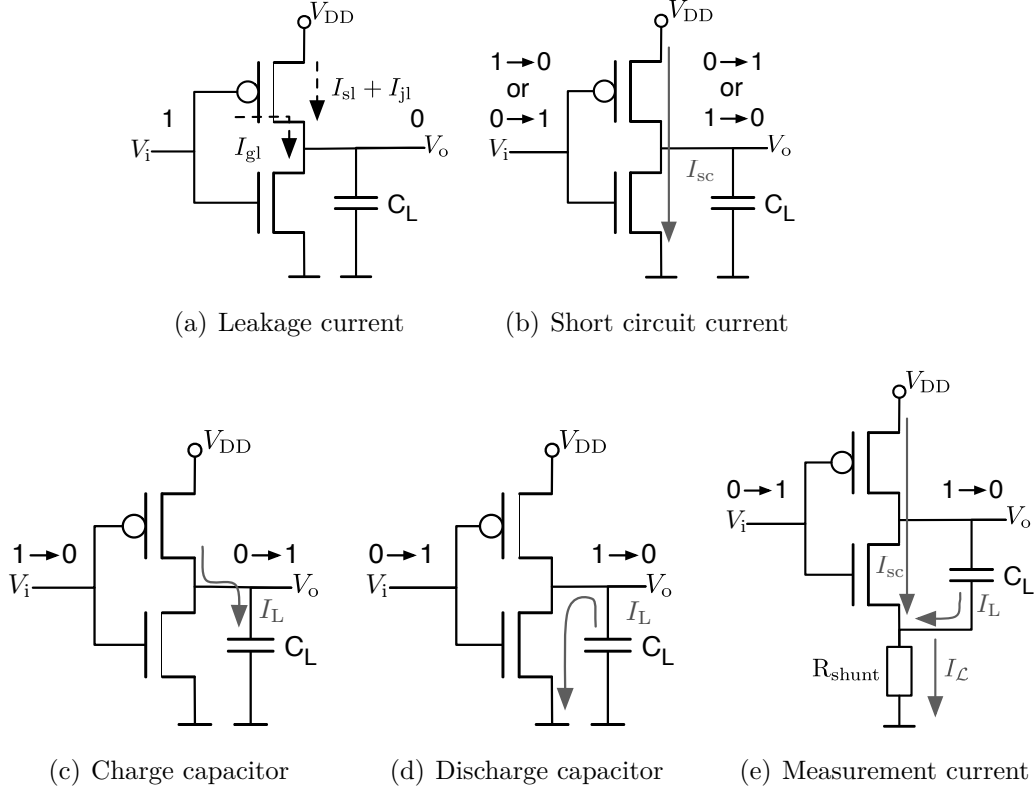


Figure 2.4: Example of power consumption on an CMOS-inverter

is floating from the source-to-body junction to the drain-to-body junction and is caused by the need for higher doping levels while the junctions at downscaled CMOS technology platforms get shallower.

Short-Circuit Power Dissipation At any bit transition on the input of the CMOS-inverter a short-circuit current I_{sc} is floating across both transistors from V_{DD} to ground.

$$\mathcal{P}_{sc} = I_{sc} \cdot V_{DD} \quad (2.7)$$

When both transistors of the inverter are short cut during the switching process a short circuit current I_{sc} will be caused, cf. Fig. 2.4(b). The mount of I_{sc} depends on the fabrication properties of the CMOS transistors, for instance the parasitic capacity C_L , as well as the rise- and fall-time of the incoming and outgoing bit transitions, cf. [Vee84]. Thus, I_{sc} increases by down scaling the CMOS technology.

Dynamic Power Dissipation Bit transitions at the input of the inverter also cause dynamic power dissipation besides the short circuit power dissipation.

In case of no bit transition only $\mathcal{P}_{\text{stat}}$ from Eq. (2.6) dissipates power. In the following the dominant influence of the parasitic capacity C_L on the dynamic power dissipation is shortly discussed. Compared to the \mathcal{P}_{sc} the bit transitions have to be distinguished from the dynamic power dissipation, cf. [WE93]. C_L is charged or discharged depending on the bit transition and thus causes a power dissipation in the CMOS circuit, cf. Fig. 2.4(c) and Fig. 2.4(d). In case of a bit transition $1 \rightarrow 0$ the P-MOS transistor is cut short. Thus, C_L is charged by I_L (as depicted in Fig. 2.4(c)), resulting in the following absolute value of power dissipation under the condition that V_o equals the logical 1:

$$\mathcal{P}_{1 \rightarrow 0} = \mathcal{P}_p = \frac{1}{T} \int_0^{V_{DD}} C_L \cdot (V_{DD} - V_o) dV_o = \frac{C_L \cdot V_{DD}^2}{2T} = \frac{1}{2} \cdot C_L \cdot V_{DD}^2 \cdot f. \quad (2.8)$$

For a $0 \rightarrow 1$ transition the P-MOS transistor is opened and the N-MOS transistor is cut short to ground. Thereby the stored energy in the capacitor C_L from the opposite transition is deployed of the N-MOS transistor to the ground. In this case the absolute value of power dissipation is calculated by:

$$\mathcal{P}_{0 \rightarrow 1} = \mathcal{P}_n = \frac{1}{T} \int_0^{V_{DD}} C_L \cdot V_o dV_o = \frac{C_L \cdot V_{DD}^2}{2T} = \frac{1}{2} \cdot C_L \cdot V_{DD}^2 \cdot f. \quad (2.9)$$

The overall power dissipation of the CMOS inverter is sum of both transitions for one cyclic transition at the operating frequency f :

$$\mathcal{P}_{\text{dyn}} = \mathcal{P}_p + \mathcal{P}_n = C_L \cdot V_{DD}^2 \cdot f. \quad (2.10)$$

If the switching activity α for each clock cycle is not 100 % then

$$\mathcal{P}_{\text{dyn}} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot f. \quad (2.11)$$

Exploitable Power Consumption Since power analysis attacks exploit data-dependent power consumption, the power dissipation or consumption during the switching activities are of great interest. As introduced in Subsect. 2.2.1, the exploitable power consumption $\mathcal{P}_{\mathcal{L}}$ is measured from the device directly over an ohmic shunt R_{shunt} , which is usually placed between the ground line and the active device, cf. Fig. 2.4(e). The voltage drop $V_{\mathcal{L}} = I_{\mathcal{L}} \cdot R_{\text{shunt}}$ of the shunt is then used to estimate $\mathcal{P}_{\mathcal{L}}$:

$$\mathcal{P}_{\mathcal{L}} = \begin{cases} \mathcal{P}_{0 \rightarrow 1} + \mathcal{P}_{\text{sc}} \approx \frac{V_{\mathcal{L}}^2}{R_{\text{shunt}}} = (I_{\text{sc}} + I_L)^2 \cdot R_{\text{shunt}} & 0 \rightarrow 1 \text{ input transition} \\ \mathcal{P}_{\text{sc}} \approx \frac{V_{\mathcal{L}}^2}{R_{\text{shunt}}} = I_{\text{sc}}^2 \cdot R_{\text{shunt}} & 1 \rightarrow 0 \text{ input transition} \end{cases} \quad (2.12)$$

The exploitable power consumption of the inverter depends not only on any bit transition, but it also depends on a rising edge bit transition or the opposite

transition. In case the transition goes from 1 to 0 according to Fig. 2.4(c) C_L is charged and only I_{sc} is the current source of I_L floating through R_{shunt} . While in case of a $0 \rightarrow 1$ transition I_L is the sum of I_{sc} and I_L stored in the capacitor, as depicted in Fig. 2.4(e). Please keep in mind that Eq. (2.12) only describes the power consumption over an ohmic shunt in conjunction between the device and the ground line. If the shunt is placed in the supply line, then the equation changes, cf. [GHP04]. The first verification on the observation in [GHP04] on different micro controller platforms was done in [PSQ07].

2.2.1.2 Attack Methods

The various power analysis attack methods that have been introduced in the recent years and studied in the side-channel community can be separated into two groups. The attack scenario as well as the strength of the adversary defines the two different groups of side-channel attacks. The first group contains all the non-profiling attacks, in which the adversary has limited access to the targeting device in terms of time and functionality. He or she is only able to monitor the power consumption of the implementation and has access to the public data, for instance the ciphertext or plaintext, for each cryptographic operation performed by the device.

The second group contains all the analysis attacks, which include a training phase with a physically and functionally identical device. Thus, the so-called profiling based attacks are mostly separated into two attack phases. In the first phase the adversary is constructing patterns, often called templates, for expected power consumption values depending on the intermediate values or states of the implemented cryptographic algorithm. After that, in the second phase, utilizing the created templates of the first phase attacks the targeting device. Thus, the attack phase is much more efficient and therefore faster to conduct, which is of high interest in case of very limited time access to the targeting device.

Various Attack Methods Table 2.1 lists several basic power analysis attack methods, classified into the two main groups with the according references in the literature. Beside the listed attack methods in the table many derivative attacks exist in literature, thus the table should provide a basic overview of the different basic techniques used in order to exploit the power consumption. The basic methods differ in terms of their statistical or stochastic techniques of the distinguisher², the exploited power model, or the focus of exploitation (control flow or intermediate operations).

For instance, the simple power analysis (SPA) attack exploits unbalanced

²The distinguisher in the attack phase cf. paragraph **Estimated Power Consumptions**

Table 2.1: Power analysis attack methods

Type	Name	Comment	Reference
Non-profiling	Differential power analysis	First attack on the intermediate values	[KJJ99]
	Correlation power analysis	Most popular attack method	[BCO04]
	Mutual information analysis	Considers non-linear dependencies	[GBTP08]
	Linear regression analysis	Utilizes linear regression for modeling	[DPRS11]
	Power analysis collision	Exploits intermediate collisions	[MME10]
	Power Amount Analysis	Considers the state-wise power amount dissipation	[THH12]
Profiling	Simple power analysis	First control flow oriented attack	[KJJ99]
	Template attack	Utilizing power consumption densities	[CRR02]
	Stochastic approach	Exploits a subspace representation	[Sch05]

control flow of public cryptography schemes like RSA, ECC, or McEliece³, cf. [KJJ99, SSMS09, MSSS11]. The different power consumption value patterns of a captured \mathbf{p}_t -vector are directly interpretable due to the known control flow decisions in the algorithm. The other two attack methods, listed in the table in the section profiling, exploit the intermediate values of internal operations and thus exploit the information of many \mathbf{p}_s -vectors. The stochastic approach differs from the template attack by using a more advanced power model based on subspaces and linear regression.

All the attacks sorted in the section *non-profiling* in Tab. 2.1 are also focusing on the data-dependent power consumption of the internal operations like the stochastic approach and the template attack. The first four attack methods, from top to bottom, exploit the switching activity within the circuit and only differ in terms of used distinguisher functions. While the differential power

³ A post quantum cryptography scheme [SWM⁺10]

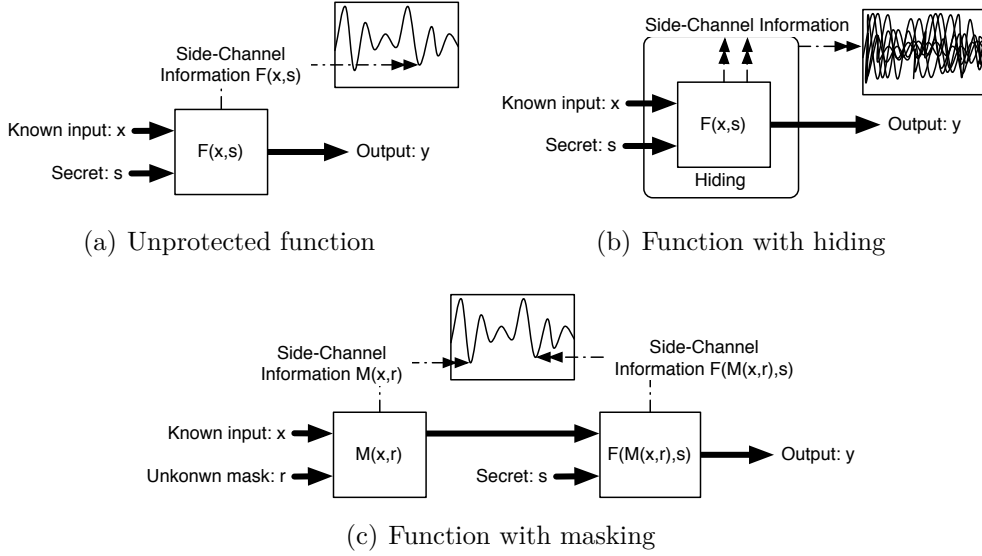


Figure 2.5: Countermeasure methods for an example function

analysis (DPA) attack and the correlation power analysis (CPA) attack only exploit linear relations, the mutual information analysis (MIA) attack and the linear regression analysis attack can exploit non-linear relations. The power analysis collision attack does not exploit any internal switching activity of the implementation compared to the other non-profiling attacks. This attack exploits the significant identifiable power consumption of an internal collision of two sequentially identical data values. The adversary injects the collisions of the intermediate values by chosen-plaintext scenario.

The Power Amount Analysis utilizes the concept of the *Additive Withe Gaussian Noise* model known from the area of communication engineering to more efficiently extract information out of the captured power traces. The methods compress a time frame of each capture power traces into one value by calculating the variance of this frame over the time. According to the assumption of an stationary and ergodic process the additive noise in the power trace has a constant value, which is thereby separated from the dynamic exploitable data-dependent power consumption.

2.2.2 Countermeasures against Power Analysis Attacks

Since the discovery of the outgoing threat of such power analysis attacks countermeasures have been studied and developed. An overview of the general concepts of countermeasures against side-channel analysis attacks will be given in this subsection. In order to lower the success rate of such an attack, an intuitive approach is to decrease the exploitable physical information leakage, which is

emitted while the implemented algorithm is active. The exploitable information leakage can be decreased mainly by two different basic approaches. The first approach, *hiding*, focuses on the physical behavior of the implementation. The principles of hiding methods are to decouple the data specific power consumption from the actual internal processed intermediate values, cf. Fig. 2.5(b). The second approach, *masking*, reduces the degree of exploiting the data-dependent power consumption by internally randomizing the processed intermediate values by calculating with auxiliary or masked values instead, cf. Fig. 2.5(c).

Masking In the area of side-channel countermeasures the principle of masking refers to the concept of secret sharing. The incoming public as well as the intermediate values are masked by a randomly chosen value, the so-called masking value. Hence, the internal operation processes are unknown and thus it is not possible to directly derive intermediate values from the known input or output data. In order to achieve the correct output as in the non-masking case the intermediate results have to be corrected at the end of the crypto algorithm.

Without knowing the mask value the adversary cannot construct correct hypotheses about the estimated power consumption depending on the known input or output of the cryptographic algorithm, as depicted in Fig. 2.5(c). Therefore, the focusing side-channel information in the unprotected case, as shown in Fig. 2.5(a), differs from the masked case. Nevertheless, if the adversary combines both side-channel informations of the masked circuit, cf. Fig. 2.5(c), he can exploit the power consumption by a so-called second order attack. Thereby, the attacking effort for conducting a successful attack on a masked circuit compared to an unmasked circuit increases by the factor of $2^{\text{number of masking bits}}$.

Over the recent years different masking methods have been developed for block cipher algorithms as well as for asymmetric encryption schemes. Due to the algorithmic nature of the masking concept, i.e., secret sharing, this kind of countermeasure is mainly implemented on the algorithmic layer of an embedded system and utilizes the existing hardware architecture. Nevertheless, in chip design some side-channel resistant logic styles have been developed to apply the concept of masking on cell level as well, but at a high cost of area consumption. Tab. 2.2 provides a basic overview of the most commonly used masking techniques listed in the literature.

Hiding Compared to masking the hiding concept references to counterfeit the leakage exploitation on the physical level of the implemented circuit of the cryptographic algorithm. The unmodified intermediate values, which rely on the input values, can be decoupled from their characteristic power dissipation in two domains: the time domain and the amplitude domain. Due to the implementa-

Table 2.2: Countermeasures Methods

Type	Name	Level	References
Masking	Arithmetic masking	Algorithm	[GT02, TSG02]
	Boolean masking	Algorithm	[HOM06, CB08]
	Affine masking	Algorithm	[FMPR10]
	Masking logic	Cell	[TKL04, GrM04, FG05]
	Blinding	Algorithm	[Cor99, JT01]
	Masked Dual-Rail Pre-Charge Logic (MDPL)	Cell	[PM05, PKZM07]
	Threshold implementation	Algorithm	[NRR06, NRS11, MPL ⁺ 11]
Hiding	Dummy operations	Algorithm	[CF05, MPO07]
	Shuffling	Algorithm	[MPO07, MSH09, SMH09]
	Random delay, clock	Cell	[LOM10, LBHO10]
	Dual-Rail logic	Cell	[TV04, VK11, HdLTR12]
	Triple-Rail logic	Cell	[LMT ⁺ 09]
	Noise, Random switching	Cell	[SSI04, GM11]
	Register pre-charging	Algorithm	[LBHO10]

tion of the circuit the signal-to-noise ratio of the exploitable power consumption to the remaining consumption at one point in time decreases, cf. Fig. 2.5(b).

Increasing the noise of the unwanted power dissipation is one possibility to decrease the signal-to-noise ratio. Thereby the amplitude of the exploitable power consumption is included in the other power consumption by manipulating the amplitude or shuffling the operation time to achieve a lower in the background noise embedded average level. The second segment of Tab. 2.2 refers to some countermeasures, which can either be applied on the algorithm level or on the cell level of a design.

The second possibility to reduce the exploitable power consumption is to balance the circuit in a manner that always the same power consumption is needed, regardless of the processed data. The challenge of achieving such a physical behavior, a circuit with two complementary parts with identical placement and routing on the cell layer of the design is required. Such tools for placement and routing are described in [YS07, HdLTR12]. In custom ASIC design the design goals are easier to achieve than on a FPGA platform due to the full control over the silicon surface. Another method to balance the power consumption is proposed in [GM11] by leveling the switching activity.

Concept of Mutating Runtime Architectures

Contents

3.1	Principles of Mutating Data Paths	24
3.2	Online Allocation:	
	Different Types of Processing Units	29
3.2.1	Influence on the Power Signature	30
3.2.2	Requirements for the Type of Process Unit Selection . . .	33
3.2.3	Methods of Randomizing the Processing Units	36
3.3	Dynamical Binding:	
	Degree of Parallelism	42
3.3.1	Impact of Non-Static Parallel Processes on the Noise . . .	43
3.3.2	Questions on how to Establish a Randomized Concurrency	47
3.3.3	Randomized Concurrency by Virtualization	51
3.4	Flexible Scheduling:	
	Progression of Tasks	56
3.4.1	Effect on Power Consumption Patterns	57
3.4.2	Partitioning the Algorithm	59
3.4.3	Randomized Micro-Program Selection	61
3.5	Design Flow of Mutating Data Paths	63

The goal of hardening a design with countermeasures against implementation attacks is to lower and to minimize the amount of exploitable physical information. Thus, the utilized technology platform as well as the architectural structure of the design have to be modified in order to reduce the exploitable physical information of the implemented circuit. The key concepts of this thesis topic will be introduced and discussed in theory in this chapter. First, the fundamental considerations on the mode of action of mutating data paths with respect to countermeasures against power attacks are analyzed. In this context it is discussed,

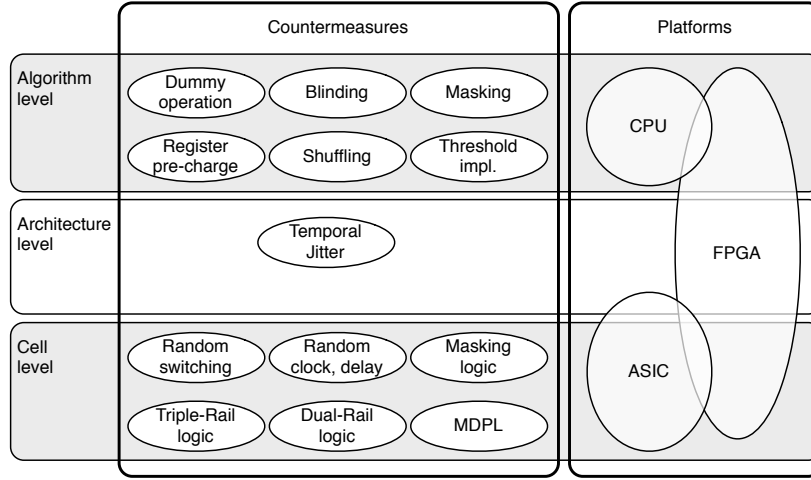


Figure 3.1: Overview of countermeasure techniques and related platforms

which components and properties should be randomized on the architectural level in order to achieve a higher randomization of the power consumption.

3.1 Principles of Mutating Data Paths

Most countermeasures against power analysis attacks are either applied on the algorithm level or on the cell level of the implemented cryptographic algorithm, cf. Subsect. 2.2.2. The reason for this circumstance is schemed in Fig. 3.1: The developed countermeasures are always bound to the platform, on which the security sensitive application is running and has to be protected. Therefore, it sounds reasonable that most countermeasures are developed on the algorithmic level or on the cell level. In case of protecting a software implementation on a standardized platform with a fixed CPU core, the countermeasure can only be implemented algorithmically in software. In case of a full-custom chip (ASIC) design, for instance a smart card chip, the designer has the freedom to integrate countermeasures on the cell level. In both cases the physical architecture of the implementation is considered to be fixed during runtime, due to the platform properties of an ASIC or a general CPU. Compared to that, the concept of a FPGA platform offers more flexibility, because of its approach of runtime reconfiguration. In the context of side-channel countermeasures only one approach has been published so far which utilizes the concept of changing the architecture during runtime. The so-called *Temporal Jitter* countermeasure was proposed in [MGV08] by Mentens et al. and utilizes the reconfiguration of the data path in order to freely position the register and the combinatorial logic in the circuit.

Thereby, the activity of each clock cycle is altered and, similar to shuffling, the adversary does not know when which operation is being processed. The data path structure and the design architecture can also be changed dynamically in order to improve the side-channel resistance. In the following, different methods of mutating the architecture in terms of the data path are discussed, as well as methods to construct mutating data paths.

Freedom of Design Process In recent years, the research area of computer aided design has developed different methods and techniques, as well as basic steps to map an abstract algorithm to an integrated circuit design. The concept of high level synthesis on processor level is well-known from the area of computer aided design (CAD) and contains the following three basic design process steps in order to implement an algorithm:

- Allocation
- Binding
- Scheduling

The necessary basic operations to perform the algorithm are identified in the basic design process step *allocation*. Each basic operation is then mapped to one or multiple types of hardware processing elements, which can generally perform the operation. The assignment of a specific task of the algorithm, an operational request at a certain point of the algorithm execution, is done by the basic design process step *binding*. The specific assignment of a task of the algorithm to a hardware processing element is done by this design step in general. Therefore, no additional information is necessary when the task has to perform the operation. The basic design process step *scheduling* is used to schedule the tasks. Thus in this step the designer can decide how many operations are done in parallel or in which sequence independent operations are performed.

The architectural properties of the circuit directly rely on the results of the above mentioned basic design process steps of a CAD process, cf. [GAGS09]. Thereby, the physical characteristics of the implemented algorithm, for instance the execution time, the power consumption, or the area consumption, can be influenced by each of the three steps. Additionally, some decisions in one of the three basic design process steps, for instance binding, will affect the other two basic design processes as well (in this example allocation and scheduling).

The principle of mutating data paths utilizes the influence of the basic design process steps on the architecture in order to create a flexible design space exploration. This exploration contains various implementations of the same cryptographic function with different physical properties. During runtime, the different

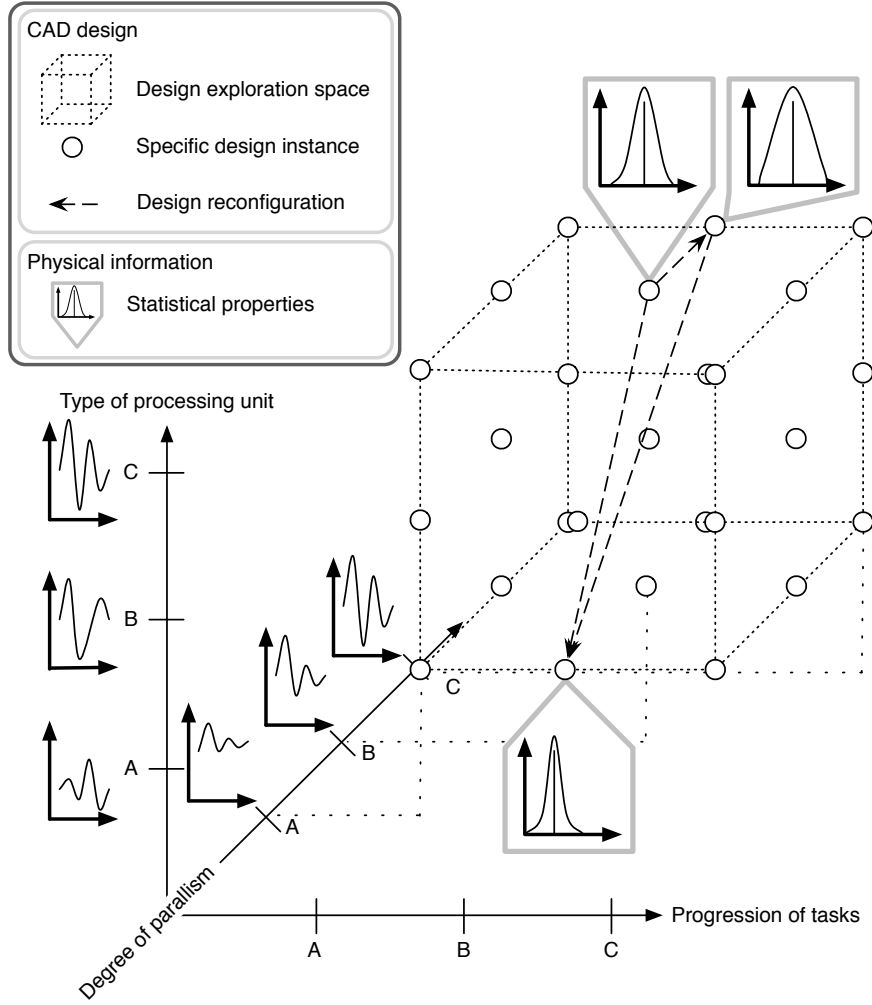


Figure 3.2: Modularization of the design exploration space

designs should be accessible within the design exploration space from any position, cf. Fig. 3.2. The method to mutate from one design to another is possible by an active reconfiguration process. The granularity of the reconfiguration has to be at a partial level, so that design properties $\mathfrak{dp}(\cdot)$ can be changed individually by the settings of the corresponding basic design process step exclusively. Thereby, the physical behavior of the design, for instance the statistical characteristic of the data-dependent power consumption, can be manipulated directly, while the implementation maintains the same functionality.

The arising challenge from this mutating approach is to identify design properties $\mathfrak{dp}(\cdot)$ which can be controlled by the corresponding basic design process steps individually or almost individually. Therefore, the reconfiguration scheme between the designs in the exploration space can be interpreted as a path through

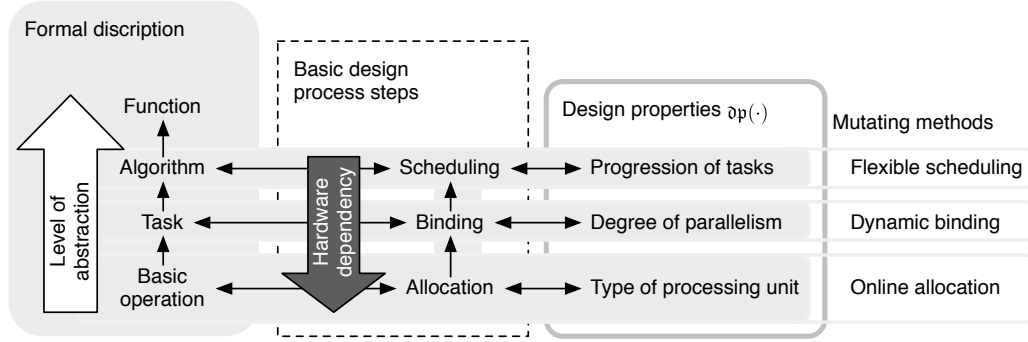


Figure 3.3: Layer model for modularization

the design exploration space. Figure 3.2 visualizes the interconnection of different modularized designs by slashed arrows inside the design exploration space. Therefore, the architecture has to have a very modularized structure in order to manipulate only the intended $\mathbf{dp}(\cdot)$.

Modularization of the Design Exploration Space Usually, the CAD-based basic design process steps are used to transform a formally described function to a fixed and optimized design with a certain $\mathbf{dp}(\cdot)$. Due to the fundamental principle of the mutating data path concept the exploration space has to be modularized concerning the dependency between the basic design process steps. The first step is to identify flexible design parameters that influence the physical properties of the circuit, but are nearly independent from each other. These design parameters are then grouped together into clusters of $\mathbf{dp}(\mathbf{pv})$ with different sets of parameter values \mathbf{pv} . Therefore the combination of different values of each design property leads to a design with new physical features. The chosen design properties thereby orientate on the classical development flow from a formal description of function to the $\mathbf{dp}(\cdot)$ of the implemented circuit. As visualized in Fig. 3.3, the known basic design process steps in the area of CAD rely on each other in general. For the modularization of the design space certain independent $\mathbf{dp}(\cdot)$ can be derived from the design process step by utilizing different methods, cf. Fig. 3.2 and Fig. 3.3. Each of the proposed methods is used to create different partial designs regarding the requirements of the design properties. The second request to the chosen $\mathbf{dp}(\cdot)$ is that they should specifically change the physical behavior of the design. Each design property should influence the power consumption characteristics significantly, so that each property affects the statistical behavior of the monitored power consumption signature. As depicted in Fig. 3.2, the proposed selection of $\mathbf{dp}(\cdot)$ manipulates the variance

$\sigma_{\mathcal{P}}^2$ and the mean $\mu_{\mathcal{P}}$ of the exploitable power consumption $\mathcal{P}_{\mathcal{L}_t}$ at time instant t . In general, the connection between these statistical characteristics of the power consumption signature and the $\mathfrak{dp}(\cdot)$ can be expressed by:

$$\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\} \leftarrow \mathfrak{dp}(\cdot). \quad (3.1)$$

Based on the assumption of the strongest theoretically possible attack (template attack, cf. Subsect. 2.2.1.2), where the power consumption is deterministic and the noise of the power consumption is a multivariable Gaussian process, the distribution properties μ and σ^2 are the focusing properties in the here discussed concept.

Selected design properties The result from the basic design process step allocation is the $\mathfrak{dp}(\text{processing unit})$. It specifies which *type of processing unit* is used in the algorithm. The type of the processing unit inherits its parameter from the process step allocation, but is nearly independent from the process steps binding and scheduling. Thus, it is possible to change the processing unit by the method *online allocation* during runtime without changing the bound task to neither the unit nor the execution time in the task sequence.

The second identified $\mathfrak{dp}(\text{concurrency})$ is the *degree of parallelism*. It inherits its design parameters from the process step binding. Independent from the type of the allocated processing unit the degree of parallelism can be varied based on the amount of available required processing units. The dependency to the other $\mathfrak{dp}(\cdot)$ is inherited from the basic design process, thus scheduling is not as completely independent as binding or allocation. The degree of parallelism can be only applied if the proposed method *dynamic binding* process does not violate any task dependency.

The last $\mathfrak{dp}(\text{task})$ is connected to the basic design process step of scheduling and it defines the *progression of tasks*, which are sequentially executed due to the task dependencies. The data flow of the sequential or concurrent conducted tasks is independent from the utilized processing unit on the hardware layer from a functional point of view. The dependency to $\mathfrak{dp}(\text{concurrency})$ can be handled with additional constraints and considerations. Thus, sequentially aligned and similar tasks may be parallelized by $\mathfrak{dp}(\text{concurrency})$ and therefore have to be considered especially in the proposed method *flexible task*. In this work the flexible task managing is coped by a middleware approach and micro-machine based coding for the control flow. The details to this approach will be detailed in the corresponding section.

The partitioning for a flexible hardware-/software codesign sounds reasonable on the FPGA platform because of the constraints between $\mathfrak{dp}(\text{concurrency})$ and $\mathfrak{dp}(\text{task})$. Managing the task progression is more flexible and easy to handle in

software as well as the dynamic binding for the exploited degree of parallelism of the processing units. Only the online allocation of the different types of the processing units cannot be done in software due to its hardware dependency.

The modularization of the exploration space can be achieved by two steps: First, identify design properties $\mathfrak{dp}(\cdot)$, which can be gained from the basic design process steps. Note that the selected $\mathfrak{dp}(\cdot)$ due to their design parameters are almost independent from each other. Second, separate the design into partitions in order to perform a more flexible hw/sw-codesign.

In the following section the different $\mathfrak{dp}(\cdot)$ and the proposed methods to utilize the $\mathfrak{dp}(\cdot)$ will be discussed in detail. The effect of the different $\mathfrak{dp}(\cdot)$ to the physical behavior of the circuit will be discussed in general, as well as in terms of side-channel analysis. For each of the three vectors, which span the design exploration space, a metric will be introduced in order to verify the setting of the parameter values \mathbf{pv} of each $\mathfrak{dp}(\cdot)$. Thus, Eq. (3.1) is refined in:

$$\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\} \leftarrow \mathfrak{dp}(\text{processing unit}, \text{concurrency}, \text{task}). \quad (3.2)$$

3.2 Online Allocation: Different Types of Processing Units

The design property type of processing units may be interpreted as an online allocation of the same maintaining functionality, but executed on a different implementation. The processing unit in this context is a basic operation, which is exploited by a superior algorithm. The processing unit is a hardware component (usually an IP-core) of the implemented cryptographic function. It provides a requested functionality to the superior algorithm of the implemented cryptographic function.

General considerations The algorithm exploits only the deterministic behavior of the processing element at an abstracted level of an event based function $f(\cdot)$. The function transforms a given input set of intermediate values x_{input} into an output set of intermediate values x_{output} in the algorithm flow:

$$x_{output} \leftarrow f(x_{input}) \quad (3.3)$$

The physical behavior of the processing unit is not considered in the abstract formalism of the algorithm flow. Neither the power consumption nor the processing time is a constraint in the algorithmic description of the cryptographic algorithm. Therefore it can be modified without changing the algorithm, from the point of prospective of the allocation step. Please note, the handling of the correct executions of tasks or the point in time of the requested basic operation is not in the focus of the allocation process. In embedded devices not every basic operation, which is utilized in the superior cryptographic algorithm can be altered, due to hardware constraints. A fundamental request on the selection of the basic operation to be altered is that a broad variance of implementations exists so that the influence of the physical behavior can be assured. Also a certain implementation complexity of the selected processing unit has to be considered in order to have significant change in the power consumption signature. The influence of the selected processing unit to the power consumption of the implemented cryptographic algorithm has to be a major part of the total power consumption.

In the following Subsection the effect on the power consumption signature by altering the implementation of the basic processing unit will be introduced and discussed. Therefore, the origin of the different power consumption signatures will be detailed as well as examples will be highlighted to provide an impression on the effect of the online allocation. Also a metric will be introduced to support a designer to evaluate, if the different implementations of the selected processing unit are working beneficially together in order to increase the noise and randomness. At the end of this section different methods for an online reconfiguration of the processing unit will be introduced.

3.2.1 Influence on the Power Signature

The power consumption of CMOS technology based circuits relies on the switching activity of the circuit. As introduced in Subsect. 2.2.1.1 the dynamic power consumption as well as the power consumption based on the short-circuit current can be exploited for side-channel attacks. Therefore, data-dependent glitches have a significant impact on the exploitable data-dependent power consumption. In complex circuits the combination and interconnection between the basic boolean operations are the cause for the circuit data-dependent characteristic glitches. The combinatorial part of the circuit generates the glitches due to different run-time delays of the individual Boolean function blocks and the so-called early propagation [HdlTR12] effect of the Boolean function blocks. The switching probability of the most atomic unit of the utilized hardware platform, for instance a gate or a LUT, is an indicator for the amount of glitches, which may be propagated through the combinatorial part of the circuit.

On the FPGA platform the interconnection and the configuration setting of the LUT are causing the glitches during the transient phase of the combinatorial part of the circuit. The different multiplexer components on the Virtex 5 platform are designed with pass-gate transistor circuits and thus do not cause glitches, cf. [ug1].

Different implementations of the utilized basic operations within the cryptographic algorithm will influence the exploitable power consumption signature by its switching activity. The various internal architectures, interconnections, and configuration settings of the LUT of the FPGA are the origin of the characteristic controlled transient effects or glitches.

Considerations about the power analysis The statistical properties of the data-dependent power consumption in terms of the mean and the variance is thereby strongly influenced by the characteristics of the glitches of the corresponding circuit. In case of a static cryptographic algorithm design this property is relatively vulnerable to side-channel attacks, cf. [MPO05, MPG05, FG05, SHAS09]. The attacks exploit the unique distribution of the power consumption signature, which is caused by a set of equally distributed input parameters to the attacked implementation. One possibility to compare the power consumption signature of different designs is to use a scatter plot like in Fig. 3.4

Example of different SBox designs The scatter plot in Fig. 3.4 clearly visualizes the different power consumption signatures of different SBox designs. Each of the analyzed circuits consists of one 8 bit register and one of the four different 8-bit AES SBox designs. For the experiment each design was fed with the same input data and the power consumption was captured by the same FPGA evaluation board (SASEBO). The plot compares pairwise the power consumption of different SBox designs. On the x-axis the power consumption value at time instant t of one SBox design A is displayed. The y-axis marks the power consumption value of SBox design B. If both designs are completely identical (same implementation for design A and B) the plotted points of this scatter plot should be distributed on a diagonal line through the origin of the plot. The diagonal line is schemed by a dashed line for reasons of clarification.

Based on the plot results in Fig. 3.4 the designs obviously differ from each other. The scatter plot of the four different SBox designs in Fig. 3.4 clearly shows which designs are most similar to each other. The comparison between

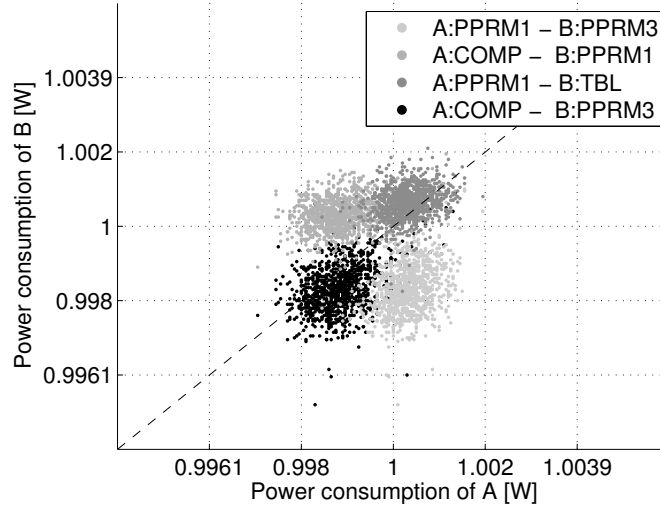


Figure 3.4: Scatter plot of the power consumption of different design

the SBox designs *PPRM1* and *TBL*, as well as *PPRM3* and *COMP*, seems to be more similar than the other two comparisons, because some of the points of their scatter plots are on the dashed diagonal line. The SBox designs *PPRM1* and *TBL* are both lookup-table based implementations. The only difference between these designs is the Boolean representation of the SBox function. Therefore, the distribution of the power consumption values of these two designs are in some cases more similar. While in case of the design *TBL* every Boolean expression was used to implement the SBox on the FPGA, for the design *PPRM1* only the Boolean operations *XOR* and *AND* were used. The same holds for the SBox designs *PPRM3* and *COMP*. In their case both designs are based on a composite fields architecture, but utilizing the same field. Again, the only difference are the Boolean expressions, which were used to implement the SBox function. The *PPRM3* was in this case the constraint design, which only utilizes *XOR* and *AND* as basic Boolean operations.

This experimental example clearly indicates the difference in the exploitable power consumption signature of the four different AES SBox designs. Both properties, the LUT configuration and the interconnection architecture of the LUT on the FPGA platform cause a different distribution of the power consumption. The distribution strongly depends on the switching activity, which is based on the data-dependent transitions and glitches in the implementation circuit.

Implication for the selected design Processing units with different power consumption characteristics should be selected in order to minimize the ex-

exploitable power consumption. Therefore, the different characteristic power consumption distributions of the selected processing units should be joint together into one big complex power consumption distribution. The more similar the distribution of the different processing unit types are in terms of $\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2$ and $\mu_{\mathcal{P}_{\mathcal{L}_t}}$ ¹, the less complex is the joint distribution. By selecting different processing units with a wide-spread $\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2$ and multiple $\mu_{\mathcal{P}_{\mathcal{L}_t}}$, a randomization switching between the processing units will cause a harder characterizable complex joint distribution. Especially for profiling based attacks, like the template attack [CRR02], this method will slowly decrease the success of an attack. Thus, there is a need for a metric, which evaluates how good different designs can be grouped together in order to minimize the exploitable information about the characteristic power signature. The disadvantage of the scatterplot approach is that it has to be analyzed visually and therefore there is no direct metric that supports a quantitative comparison of different design characteristics.

3.2.2 Requirements for the Type of Process Unit Selection

As mentioned above the stronger the distributions differ for the same computed set of data the better is the randomization. A well known metric to compare distributions based on random variables is the *Kullback-Leibler divergence*, cf. [KL51]. Please note that the power consumption of the circuit can certainly be interpreted as a realization of random variables, because due to the measurement procedure and the therefore existing measurement noise. Also, the input of the cryptographic function is randomly selected and thus the intermediate values are also realizations of a random variable. The discrete Kullback-Leibler divergence is defined as follows:

$$D_{kl}(p, q) = \sum_{x \in X} p(x) \cdot \log_2 \frac{p(x)}{q(x)} = \sum_{x \in X} p(x) \cdot \log_2(p(x)) - \sum_{x \in X} p(x) \cdot \log_2(q(x)) \quad (3.4)$$

and provides the information how distinguishable the two discrete probability distributions p and q are. One disadvantage of the Kullback-Leibler divergence is that it is not a symmetric metric $D_{kl}(p, q) \neq D_{kl}(q, p)$. Therefore, it is not suitable for a distance metric for the purpose of comparing the different power consumption distributions of different processing units in practice.

Jeffery divergence An alternative symmetrical distance metric to the Kullback-Leibler divergence is the *Jeffery divergence* [Jef46]

$$D_j(p, q) = \sum_{x \in X} (p(x) - q(x)) \cdot \log_2 \frac{p(x)}{q(x)}. \quad (3.5)$$

¹Under the assumption of a Gaussian distributed power consumption

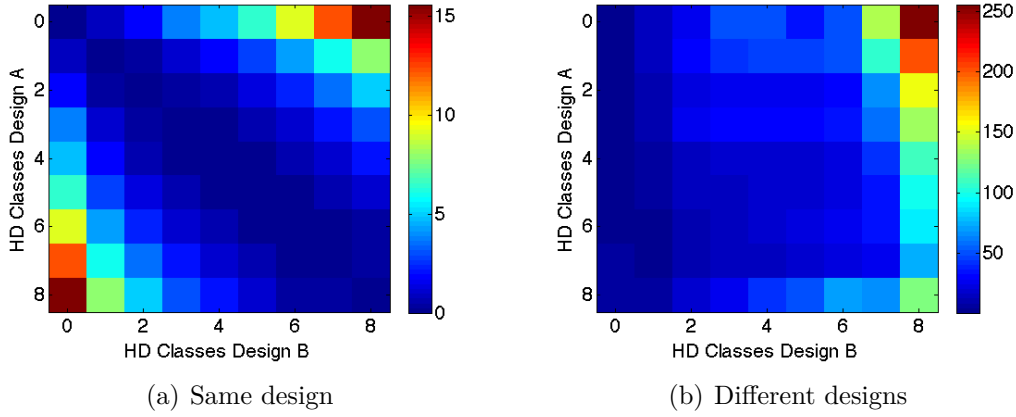


Figure 3.5: Jeffery divergence as a metric to distinguish distributions

It is a symmetrical non-zero distance metric. The smaller the value of this distance is, the more similar are the distributions. The two distributions are identical, if the result of the Jeffery divergence is zero, cf. [Jef46]. Figure 3.5 exemplary depicts the Jeffery divergence of different power consumption distributions sorted by the Hamming distance classes. In case the Jeffery divergence has been taken for evaluating the power consumption distribution two 100% identical designs, a diagonal line across the Hamming distance class establishes, cf. Fig. 3.5(a). Complementary, is the result in the Figure 3.5(b) visualizing the Jeffery divergence across the different Hamming distances of two different designs without such diagonal characteristic. The Hamming distance classes of both designs are not directly mappable due to the distribution structure of their characteristic power consumption. In some cases a mapping of different Hamming distance classes seems more likely. For instance, Hamming distance class 0 of design A is more likely similar distributed than the Hamming distance class 8 of design B. The disadvantage of the Jeffery divergence as a metric for selecting appropriate processing units is that the metric is not normalized, thus it is difficult to compare different designs with each other properly.

Total Correlation Another known method to compare random variables is the *mutual information analysis*, cf. [GBTP08]. This method stems from the area of information theory and provides the information on how much redundancy between two random variables exists. The mutual information I is defined by:

$$I(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right). \quad (3.6)$$

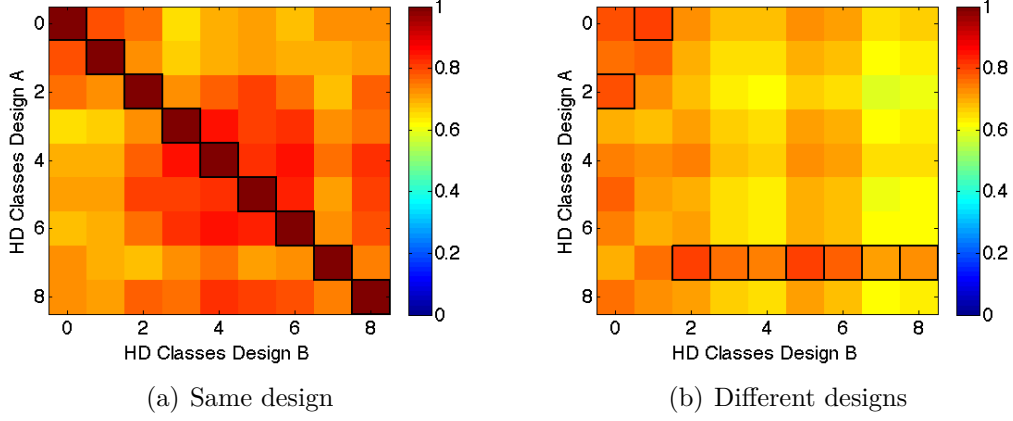


Figure 3.6: Total correlation as a metric to distinguish distributions

This symmetrical metric utilizes Shannon's entropy approach and thus can also be expressed by using the entropy $H(X)$ and the joint entropy $H(X,Y)$, as well as the conditional entropy $H(X|Y)$:

$$\begin{aligned}
 I(X,Y) &= H(X) - H(X|Y) \\
 &= H(X) + H(Y) - H(X,Y) \\
 &= H(X,Y) - H(X|Y) - H(Y|X) \\
 &= I(Y,X).
 \end{aligned} \tag{3.7}$$

The value range of the mutual information I is as follows: $0 \leq I(X,Y) \leq H(X)$ and $0 \leq I(X,Y) \leq H(Y)$. Thus it can be normalized by the the joint entropy $H(X,Y)$ in order to have a common comparable value. In 2003, Yao [Yao03] introduced the normalized mutual information

$$C(X,Y) = \frac{I(X,Y)}{H(X,Y)} \tag{3.8}$$

as an useful metric with a value range between 0 and 1, which may be used for data mining. This normalized version of I is a special case of the *total correlation* to compare two random variables.

This metric is still symmetrical due to the symmetrical property of I and provides information about the nonlinear correlation of the distribution of two random variables. A value of 1 indicates a strong dependency between the two distributions based on the analyzed random variables. Therefore, the information gained from one random variable is completely redundant to the other random variable. In case of a value of 0 the two random values have unique distributions, which share no common information. A figure of merit for the selection of

different types of processing units, in order to generate a complex distribution, is a very low total correlation value.

Figure 3.6 depicts the total correlation value plots of the same design comparison as seen in Fig. 3.5. The maximum values of each Hamming distance class comparison between the designs is highlighted by a frame of a black rectangle. The diagonal of highlighted segments in Fig 3.6(a) is intuitive, because of the one to one mapping of the Hamming distance classes of two identical designs. A more non-linear mapping is given in case of two different designs, which is depicted in Fig. 3.6(b). Therefore, the power consumption distribution of the Hamming distance class 7 of design A is more likely mapped to several power consumption distributions of Hamming distance classes of design B. This is very beneficial for the strength of switching randomly between design A and B, because this will lead to a complex non-bijective mappable distribution.

A low overall total correlation value between the different designs is a necessary, but not sufficient condition. Designs with a non-bijective mapping of the Hamming distance classes or other clusterings are a sufficient condition to achieve a complex joint distribution $(\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\} \leftarrow \mathfrak{dp}(\text{processing unit}_{(1,\dots,n)}, \cdot, \cdot))$. Therefore, the design will be more resistant to side-channel attacks by randomly switching between n different processing units with those conditions.

A low overall total correlation value between different designs is a necessary, but not sufficient condition to generate a complex distribution based on switching randomly between the individual distributions of different types of processing units. A better figure of merit is the total correlation value of Hamming distance classes between different designs of a processing unit.

3.2.3 Methods of Randomizing the Processing Units

The FPGA technology offers multiple methods and techniques to implement the concept of online allocation in a practical sense. The exchange of differently implemented processing unit types can be applied on different technical layers. First, the randomization can be done by switching network level and second, on the platform level by utilizing the reconfiguration properties of an FPGA. The approaches differ in the amount of resource consumption and total power consumption as well as the implementation time due to the complexity and tool support.

One of the first approaches to exchange the processing units in order to manipulate the power consumption signature is reported in [BMM⁺03a, BMM⁺03b].

Benini et al. proposed to utilize two different implementations of a processing unit and to randomly switch between both implementations using a multiplexer. The request on the switching probability is demanded by a probability of 0.5 in order to hinder power analysis on the design. Compared to this thesis, in their contribution they do not specify the power consumption properties of both implementations in terms of statistical properties in detail.

Switching network Using a switching network, e.g., based on a multiplexer layer or a butterfly network, is the most elementary approach to ‘reconfigure’ the data path architecture. The advantage of this technique is the short amount of time needed to mutate the data path behavior in terms of power consumption. The costs for this fast ‘reconfiguration’ are the incrementation of the data path depth, which decreases the throughput as well as it increases the amount of required resources. Especially, the cost of the not used resources of one complete processing unit implementation while the other processing unit is active is noticable.

An improvement to this approach in terms of resource consumption and throughput is given in [MSH09]. In this contribution Madlener, Stöttinger and Huss demonstrate that it is beneficial to compose the processing unit based on different elementary operations. Then, instead of switching between two complete processing units to mutate the data path behavior, the randomization is performed on the elementary components of the processing units. Thereby, a lot of resources on the primitive layer of the FPGA platform can be shared and the data path depth incrementation can be controlled finer. They propose a novel multiplier type for $GF(2^n)$, which is very supportive to embed methods to hinder side-channel attacks. In this publication the basic operations are exploited in a switching network to generate various power consumption signatures independent of the processed data.

The proposed multiplier for $GF(2^n)$, the enhanced Multi-Segment-Karatsuba multiplier (eMSK), is a further development of the Multi-Segment-Karatsuba (MSK) [EJM⁺02]. The MSK generalizes the divide-and-conquer concept of the Karatsuba multiplier by not being constraint to divide a big number into a number of segments being a power of 2. A further improvement is the eMSK scheme that it combines the efficiency of the recursive Karatsuba scheme with the flexibility of the MSK scheme. The special properties of $GF(2^n)$ are exploited as well as the application of different small MSK schemes in a sub-sequential order, which is generatable for any required segmentation.

For instance, an eMSK scheme with six segments, an eMSK₆, utilizes an ordinary halving of the original width-bit number and a MSK scheme for three segments (MSK₃). First the n-bit operand is halved by the normal Karatsuba

Algorithm 1 Multiplication with eMSK $z \leftarrow x \cdot y$ [MSH09]

Require: (x, y) Bitvectors

 $n :=$ Number of basic multiplications

 $z :=$ accumulation register

 $N[\] :=$ array of all basic multiplications

if Execute first time **then**
 $N[n] := \{0, 1, 2, \dots, n\}$
for $i = 0$ to n **do**
 $r \leftarrow \text{rand}(1, n)$
 $N_{temp} \leftarrow N[r]$
 $N[r] \leftarrow N[0]$
 $N[0] \leftarrow N_{temp}$
for $i = 0$ to n **do**
 $z_{temp} \leftarrow$ Execute $N[i]$ -th basic operation

 $z \leftarrow z + z_{temp}$
 $z \leftarrow \text{reduce}(z)$
return z

scheme, after that the eMSK₃ scheme is applied in order to achieve 6 segments, such as $\frac{n}{6}$ bit width of the two multiplicative operands. This sequential procedure of the eMSK_{2*3} reduces the number of required multiplications on the segments width from 21 basic multiplications (in case of using a MSK₆) down to 18 basic multiplications. Please note that all basic multiplications are independent to each other. Thus, they can be executed randomly, due to the commutative and associative nature of $GF(2^n)$.

The ability of the out-of-order execution of the basic multiplication is an ideal property to shuffle their sequence. Compared to well-known shuffling procedures of independent operations, for instance in an AES, this proposed procedure affects the intermediate values of the complete multiplication. Therefore, the order of the operation in the sequence strongly affects the power consumption due to the reliability of the previous intermediate value, which is stored in the same big register that is utilized for the next basic multiplication. The randomization procedure of the secured eMSK version is described in Algorithm 1. This procedure causes a randomization in the time and amplitude domain of the power consumption. The varying order of the basic multiplication has a strong impact on the glitches in the combinatorial multiplier logic as well as on the bit flips in the accumulator, cf Fig. 3.7. The additional resources² to harden the multiplier against power analysis attacks increase the resource utilization by 2% only, be-

²Occupied slices on the FPGA

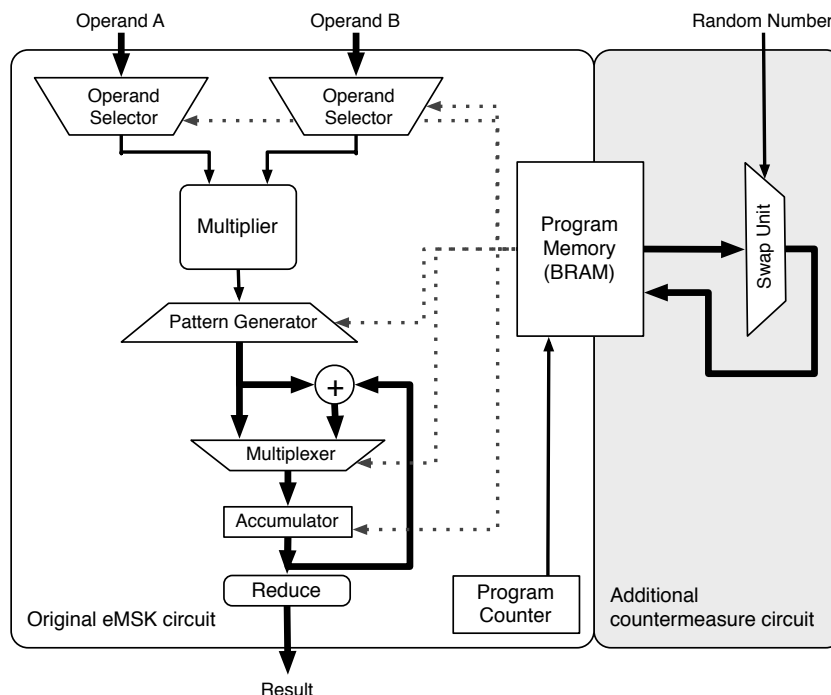


Figure 3.7: Block diagram of the hardened eMSK multiplier

cause only a swapping unit and a bigger dual-port BRAM are needed. The thesis author provided the basic idea of randomizing the sequence of elementary operations of the basic multiplications in this contribution, while Madlener optimized the multiplication scheme by combining the benefits of the recursive Karatsuba scheme with the benefits of the MSK.

A similar approach applied on the AES block cipher is given in [JSG⁺12]. In this contribution the techniques of composite field constructed SBoxes are exploited in order to switch between different SBox implementations. The principle of composite field constructed AES SBoxes is investigated and described in [Can05]. The concept of the in there presented approach is to exploit mixed bases for the composite field in order to get different intermediate representations. Compared to many other masking schemes, for instance [RYS11], the data path does not need to be doubled as in these schemes, which saves resources and makes this countermeasure suitable for resource constraint designs.

For instance, the one-element on different subfields of the composed tower fields in $GF(2^n)$ will lead to 8 different possible representations³: 0x01, 0x03, 0x05, 0x0F, 0x11, 0x33, 0x55 and 0xFF. Therefore, the one-element can be

³independent of the irreducible polynomials setting the upper bound of the subfields

Algorithm 2 Novel Masking Scheme [JSG⁺12]**Require:** 128 bit secret key k , 128 bit plaintext m , 10 random masks $Q\{q_1, \dots, q_{10}\}$ **Ensure:** 128 bit ciphertext c .

```

1: set round counter  $rc = 0$ ;
2:  $Iv = m$ ;
3: while  $rc \leq 10$  do
4:    $rk = \text{GenRoundKey}(rc, k)$ ;
5:    $Iv = Iv \oplus rk$ ;
6:   Select  $i$ -th isomorphic representation  $i = Q\{rc\}$ 
7:   Isomorphic transformation  $GF(2^8) \xrightarrow{i} GF_i(((2^2)^2)^2)$ 
8:   Inversion of  $Iv$  in  $GF_i(((2^2)^2)^2)$ 
9:   Isomorphic transformation  $GF_i(((2^2)^2)^2) \xrightarrow{i} GF(2^8)$  combined with affine
      transformation
10:   $Iv = \text{ShiftRows}(Iv)$ 
11:  if  $rc < 10$  then
12:     $Iv = \text{MixColumns}(Iv)$ 
13:     $rc = rc + 1$ ;
14:   $c = Iv$ 
15: return  $c$ ;
```

mapped to different Hamming weight classes. Such a kind of masking the intermediate values also has an impact on the data-dependent glitches in the combinatorial section of the circuit and thus, causes a different power consumption.

Algorithm 2 describes the composite field randomized masking scheme on algorithmic level. The intermediate subfield representation bases on a random number, which is used to select one of the appropriate isomorphic representations of the subfields. Please note that the isomorphic representation has to match with the utilized subfields in order to transform the intermediate value back into the normal intermediate representation after the SBox operation. Thus both transformations between the isomorphic representations in Algorithm 2, before and after the computation of the inverse into the corresponding $GF(((2^2)^2)^2)$, has to be correct. In order to save resources the transformation after the inversion can be combined with the affine transformation part of the S-box.

Stöttinger provided the basic concept of exploiting different composite field structures of the AES SBox by means of randomized switching between the subfields. The conceptional implementation and the composition of the suitable subfields was mainly done by the co-author Jungk.

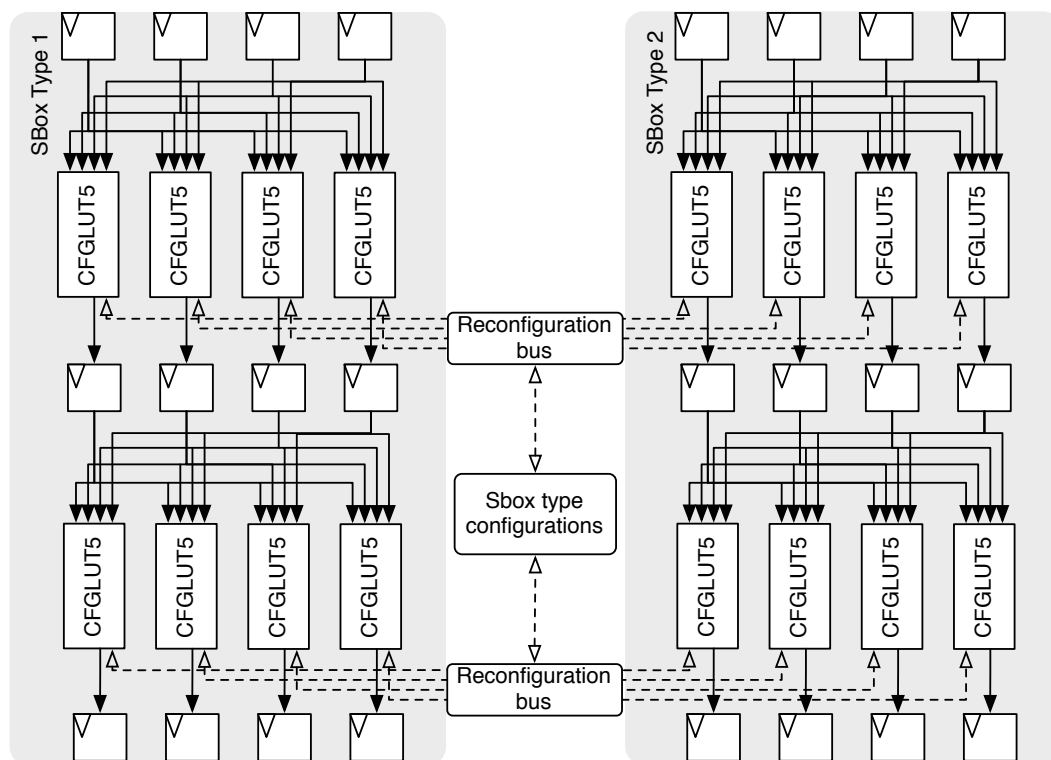


Figure 3.8: Example application for utilizing the CFGLUT5 components

Partial reconfiguration The Virtex 5 FPGA platform also offers partial reconfiguration to the designer, which can be exploited for the purpose of mutating data paths. Instead of using more active reconfigurable resources, like LUTs and routing resources, for different processing units in parallel, the partial reconfiguration can be utilized to save such resources. The data path depth incrementation is not as high as with a switching network, also the total power consumption is lower. The partial reconfiguration can be applied on two different levels on the Virtex 5 platform: on the CLB, a special Macro-Cell, and Routing level as well as on the level of the LUTs. Figure 3.15 in Chapter 2 visualizes the area and components of the FPGA that are utilized during a partial reconfiguration. The figure also depicts the necessary procedures in order to create a partial reconfigurable design. This partial reconfiguration scheme is useful if the different types of processing units have different structures, internal architectures, or utilize special macro-cells like BRAM or DSP-slices.

The Virtex 5 platform also supports a finer level of partial reconfiguration. Instead of reconfiguring a complete frame, which is the smallest reconfigurable area or number of components of the above mentioned reconfiguration scheme, the primitive CFGLUT5 can be exploited, cf. [UG612]. This primitive of the Vir-

tex 5 platform is a LUT, which is reconfigurable via a serial port. The CFGLUT5 component can be used to create any boolean function with up to 5 variables. The LUT is reconfigured with its new function within 32 clock cycles of the reconfiguration clock, which is independent from the system clock. This partial reconfiguration scheme can be used for different processing units that utilize the same architectural interconnection between the LUTs.

Figure 3.8 depicts an exemplary circuit in which the reconfiguration property of the CFGLUT5 is utilized. The circuit depicts two PRESENT SBox types, which are pipelined. The difference between SBox type 1 and SBox type 2 is the boolean functions in the LUTs, which compose the substitution function of the PRESENT SBox by the first and the second pipeline stage. Due to the different boolean functions for composing the SBox function the switching activity of both PRESENT SBoxes is different and thus, leads to different signatures.

The two PRESENT SBox modules are reconfigured by a reconfiguration bus. This reconfiguration bus interconnects each serial input and output of the corresponding LUT components of each SBox with each other. Thus, the boolean expression of one CFGLUT5 can be transferred to the corresponding CFGLUT5 of the other SBox within 32 clock cycles of the reconfiguration clock. During the transfer the functionality of the reconfiguring CFGLUT5 is not correct until the transfer has finished. In order to hide the reconfiguration time the reconfiguration clock can be driven at a higher clock rate than the actual clock rate of the system, so that the reconfiguration of the CFGLUT5 is done after the falling clock edge and before next the rising edge of the systems clock.

3.3 Dynamical Binding: Degree of Parallelism

The second design property \mathfrak{dp} (concurrency) provides another degree of freedom to span the design exploration space, cf. Fig. 3.2. This degree of freedom has a more significant impact on the physical behavior of the circuit than the influence on the functional behavior of the implemented cryptographic algorithm. The degree of parallelism influences the amount of noise due to the in parallel working process units. Therefore, the variance of the captured power consumption at a certain point in time is manipulated. Variance manipulation strongly compromises the learning phase of a template attack, in which is tried to establish a multivariable Gaussian based model for the noise distribution. In addition, the selection of different processing units working in parallel also manipulates the characteristics of the noise of the measured power consumption.

General considerations Usually during the design phase, the degree of parallel working instances is partially controlled by the binding process step. The binding also clarifies which operational task type is assigned to which type of processing unit, so that a mapping exists, which connects (not in general) a task operation with the conducting processing unit for the processing of a certain step of the implemented algorithm. After this design process step typically the binding does not change anymore. The method of dynamical binding adds a certain degree of freedom to this constraint design procedure.

One efficient method to perform a dynamic binding is by utilizing the concept of virtualizing hardware components. The virtualization is an abstraction layer, which can change the binding between a processing unit and an assigned task during runtime. Therefore, different tasks which are bound to one or multiple resources, such as processing units, can be shared or reorganized. This kind of context switching on a processing unit is handled transparently and does not need any modification of the superior scheduled sequence of tasks of the cryptographic algorithm. In conjunction with the previous described online allocation an algorithmic operation type can be executed concurrently on different types of processing units. Due to the abstract interface of the virtualization the online allocation and dynamic binding can be performed individually.

The influence of the dynamic binding of different types of processing units on the power consumption signature will be discussed in the following subsections. In specific, the influence of the virtualization process of the different types of processing units on the variance of the power consumption signature will be detailed in an example. Before the proposed virtualization method is introduced, some constraints, in order to achieve a suitable manipulation of the noise distribution of the power consumption signature by the principle of dynamic binding, will be discussed.

3.3.1 Impact of Non-Static Parallel Processes on the Noise

The influence on the characteristic power consumption is mainly contributed by the implementation of the processing unit. In order to capture the characteristic signature of the power consumption or the characteristic power consumption value of a certain value transition, multiple measurements have to be performed. On the one hand the noise is reduced or its behavior can be approximated by multiple measurements, on the other hand the average value of the expected power consumption value for a certain internal value processing can be determined.

Shuffling processing units Increasing the noise is one known appropriate method to increase the number of measurements. An increased number of measurements is necessary to determine the mean value of the power consumption

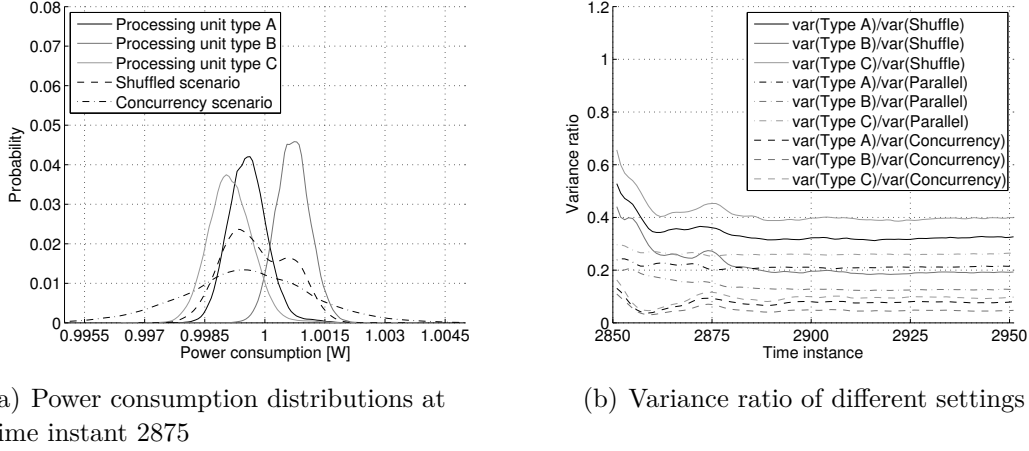


Figure 3.9: Impact of the concurrency on the power consumption distribution

$\mu_{\mathcal{P}_{\mathcal{L}_t}}$ of a certain transition. Please note that the power consumption signature of an internal operation can be characterized by the mean value under the assumption of a normal distributed power consumption of the processing unit.

One way to increase the noise level is to randomly shuffle between the different types of the processing unit implementation, cf. Algorithm 3. As depicted in Fig. 3.9(a), a shuffling between three different types of processing units increases the noise and influences $\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\} \leftarrow \mathfrak{dp}(\text{processing unit}, \cdot, \cdot)$, cf. Subsect. 3.2. Shuffling is applied by randomly binding an existing processing unit to a corresponding operational task. The distribution of the shuffled scenario $X_{\mathfrak{dp}(\text{processing unit}, \cdot, \cdot)}$ is expressed in Eq. (3.9). A measured realization of the distribution with 50,000 captured power traces is depicted in Fig. 3.9(a).

$$X_{\mathfrak{dp}(\text{processing unit}, \cdot, \cdot)} \sim \mathcal{P}_1 \mathcal{N}(\mu_{\mathcal{P}_{\mathcal{L}_t}, \text{sh}(i)}, \sigma_{\mathcal{P}_{\mathcal{L}_t}, \text{sh}(i)}^2) \quad (3.9)$$

Algorithm 3 Shuffling between m different types of processing units PU^m

Require: a given set of m different processing units types $\text{PU}^{\text{types}} = \{\text{PU}^1, \text{PU}^2, \dots, \text{PU}^m\}$ that can execute the processing unit operation op_{PU} , equal distributed random variable $R_1 = \{1, 2, \dots, m\}$ and a set of u processing unit operations, op_{PU}

- 1: **for** $1 \leq w \leq u$ **do**
 - 2: select a realization of the random variable $r_{1,w} \leftarrow R_1$
 - 3: assign each $\text{op}_{\text{PU},w}$ to the processing unit type $\text{PU}^{r_{1,w}}$
 - 4: Conduct each operation $\text{op}_{\text{PU},w}$ on the assigned PU
 - 5: **return**
-

This joint distribution in Eq. (3.9) is based on $i=3$ normal distributions $\mathcal{N}(\mu_i, \sigma_i^2)$, which are equally random selected with the probability of $p_i = 0.33$. The corresponding plot in this figure clearly depicts that the variance of the shuffling scenario is bigger than the variance of the individual distribution of the power consumptions. Nevertheless, still two out of the three mean values of the processing units $\mu_{\mathcal{P}_{\mathcal{L}_t}}$ contributions are identifiable in $X_{\text{dp}(\text{processing unit}, \cdot, \cdot)}$.

Constantly parallel execution An intuitive way to increase the variance of the distribution, the noise, is to increase the amount of parallel working processing units. Thereby the ratio between the variance of the power consumption of each i processing unit type $\sigma_{\mathcal{P}_{\mathcal{L}_t}, i}^2$ and the variance of the power consumption of the in parallel working processing units $\sum_i \sigma_{\mathcal{P}_{\mathcal{L}_t}, i}^2$ decreases. This ratio is somehow similar to a common metric in signal processing, the so-called signal to noise ratio and provides information on how good the signal is extractable. In this case it provides the information of how much the variance has increased compared to the distribution of the power consumption of each type of processing unit.

Figure 3.9(b) clearly shows that the three constantly in parallel working processing units have a bigger impact on the variance of the power consumption than the shuffling approach. Nevertheless, similar to the shuffling scenario, the variance ratio to the various processing unit types is significantly different. This fact may be exploited in order to determine which of the used processing units are working constantly in parallel. Another drawback of the constant in parallel processing approach is the additional resource consumption. If only one task has to be processed on one processing unit the other two units have to process the same data or random data, with the same statistical structure and behavior as the actual processed data. This request is necessary in order to prevent the identification of the active processing unit working on the current data and to hinder a constant statistical property of the distribution

$$X_{\text{dp}(\cdot, \text{parallel}, \cdot)} \sim \sum_{j=1}^3 \mathcal{N}(\mu_{\mathcal{P}_{\mathcal{L}_t}, j}, \sigma_{\mathcal{P}_{\mathcal{L}_t}, j}^2). \quad (3.10)$$

The distribution in Eq. (3.10) is the sum of the normal distributions of the three power consumptions due to the static parallel activity of the units.

Randomly concurrent binding An improvement in terms of decreasing the variations ratio between the randomization scenario and the processing units is the concept of concurrently active processing units. In this proposed concept the number of parallel running processing units varies as well as the binding of the operation to the processing unit. Therefore, different combinations of processing unit types, provided by the previous concept of online allocation, running with

Algorithm 4 Randomized parallel execution on n different types of processing units PU_n (concurrently)

Require: a given set of n processing units of m different types
 $PU_{\text{set}} = \{PU_1^{\text{types}}, PU_2^{\text{types}}, \dots, PU_n^{\text{types}}\}$ with $PU_k^{\text{types}} = \{PU^1, PU^2, \dots, PU^m\}$,
 $1 \leq k \leq n$ and two equal distributed random variables $R_1 = \{1, 2, \dots, m\}$,
 $R_2 = \{1, 2, \dots, n\}$

```

1: for  $1 \leq w \leq u$  do
2:   select a realization of the random variable  $r_{1,w} \leftarrow R_1$ 
3:   assign each  $op_{PU,w}$  to the processing unit type  $PU^{r_{1,w}}$ 
4: repeat
5:   repeat
6:     select a realization of the random variable  $r_{2,w} \leftarrow R_2$ 
7:   until  $r_{(2,w)} \leq \text{number of currently in parallel executing } op_{PU,w} +$ 

8:     Execute the following  $k$  operations in parallel
9:     for  $1 \leq k \leq r_{(2,w)}$  do
10:      execute  $op_{PU,k}$  with the assigned processing Unit PU
11:     $w = w + r_{(2,w)}$ 

12: until  $w > u$ 
13: return

```

a different degree of parallelism and binding together. Algorithm 4 describes the procedure of this randomization in pseudo code.

As depicted by Fig 3.9(b), the impact on the variance ratio by this principle of concurrently active processing units achieves the lowest variance ratio of all discussed randomization methods. Additionally, the ratios of the different types are grouped closer together than in the other cases, which makes it harder to distinguish, which processing type is utilized.

The joint complex distribution $X_{\text{op}(\text{processing unit}, \text{concurrency}, \cdot)}$, generated by the random concurrent binding scenario, is a composition of normal distributions of the individual power consumption of the $n=3$ utilized processing units. Equation (3.11) describes the composition procedure of that complex joint distribution $F_k(x)$. The variables i and k are used to randomly compose the distribution $F_k(x)$, thus they have to be equally distributed. While the variable i marks the selected type of the processing unit, the variable k refers to the utilized degree of in parallel working processing units. The variable j is utilized for summing up the normal distributions of the power consumption over the selected in paral-

lel running processing units. The probability of selecting a certain combination of processing units type performing the basic operation at a certain degree of concurrency is denoted by p_{ij} .

$$X_{\mathbf{dp}(\text{processing unit, concurrency, } \cdot)} \sim \sum_{j=1}^k p_{ij} \mathcal{N}(\mu_{\mathcal{P}_{\mathcal{L}_t, ij}}, \sigma_{\mathcal{P}_{\mathcal{L}_t, ij}}^2) \text{ with} \\ k \in \{1, 2, 3\}, \text{ and } i \in \{1, 2, 3\} \quad (3.11)$$

As depicted in Fig. 3.9(a), the distribution of the power consumption of the three concurrently active processing units $X_{\mathbf{dp}(\text{processing unit, concurrency, } \cdot)}$ is flatter and broader than the other distributions in Eq. 3.9 and in Eq. 3.10. Therefore, more measurements are necessary to reduce the noise and to determine the average power consumption value of a certain internal value transition. Please note that in the experiment for the concurrency running scenario the correct point in time of the targeting operation was chosen. In a real world scenario, without full control over the implementation the adversary has to perform an exhausting pre-processing phase to sort the power consumptions.

The principle of concurrently active processing units varies the degree of in-parallel working processing units and is combinable with the previously discussed method of online allocation. The combination of both methods leads to a combination of independent distributions with a broad variance, which considerably increases the effort of characterizing templates for a profiling attack.

3.3.2 Questions on how to Establish a Randomized Concurrency

There are two main questions in order to successfully conduct the concept of dynamic binding in practice. The first request encounters the question when to use the dynamical binding in the algorithm. The second request is of technical nature, in terms of implementing the dynamic binding practically. These two requests will be discussed and solutions will be presented in the following.

A reasonable and efficient execution of concurrently processed operations on processing units in terms of resource consumption needs a sufficient amount of operational tasks to be processed. Furthermore, the processed tasks have to be independent in order to execute them in parallel without compromising the intermediate values of the cryptographic algorithm. Thus, a sequence of type

Algorithm 5 Identifying processing units for random concurrency

Require: data flow graphs $DFG(V, E)$ of the cryptographic algorithm with $V = \{op_i, i=1,2,\dots,n\}$ and $E = \{\{op_i, op_j\}, i,j=1,2,\dots,n\}$, and the lower bound dp of degree in execution in parallel the processing unit operation op_{PU} on a given set of processing units $PU_{set} = \{PU_1, \dots, PU_{dp}\}$

- 1: calculate $t_{ASAP,i} = ASAP(DFG(V, E)) \forall op_i$
- 2: calculate $t_{ALAP,i} = ALAP(DFG(V, E)) \forall op_i$
- 3: calculate slack of all i processing unit operations $op_{PU,i} \in V$
 $slack_{PU,i} \leftarrow ALAP(op_{PU,i}) - ASAP(op_{PU,i})$
- 4: fill $op_{PU,i}$ from $t_{ASAP,i}$ to $t_{ALAP,i} \forall slack_{PU,i} \neq 0$
- 5: give each of the op_{PU} an unique identifier
- 6: create a resource conflict graph $G_-(V_-, E_-)$ with
 $V_- = \{op_{PU,i} \in V\}$ and $E_- = \{\{op_{PU,i}, op_{PU,j}\}, i,j=1,2,\dots,n\}$
- 7: partition $G_-(V_-, E_-)$ by applying vertex coloring to determine the chromatic number $\chi(G_-(V_-, E_-))$ by applying the *left-edge algorithm*
- 8: create m clusters cl of individual colored and with each other interconnected vertex of V_-
- 9: **return** $cl_i \forall |cl_i| \geq dp$

identical operations working on independent data is needed in order process the data concurrently. In addition, the type of identical operations should be executable on different types of processing unit implementations with varying physical behavior. In specific, with different characteristic power consumption signatures and thus different statistical properties $(\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}})$.

The request of utilizing different types of processing units concurrently is handled by the online allocation approach, while the degree of parallel execution depends on the structure of the data flow. Without rescheduling a given data flow graph (DFG) the maximum degree of parallel executable tasks for a processing unit can be determined by applying known techniques from CAD. One of the techniques to identify the data-independent, in parallel executable task vertex of processing units operation, is first to generate a resource conflict graph (G_-) and then to color the vertex of G_- , cf. [Mic94]. The chromatic number $\chi(G_-(V, E))^4$ of the graph $G_-(V, E)$ equals the number of the maximal in parallel executable identical operation types to the given data flow structure of the DFG. An efficient algorithm to determine the chromatic number χ is the *left-edge algorithm*, cf. Appendix A.2.

An extension to the above described technique to determine the maximum

⁴Minimal number of used colors to color the vertex so that each adjacent vertex has a different color.

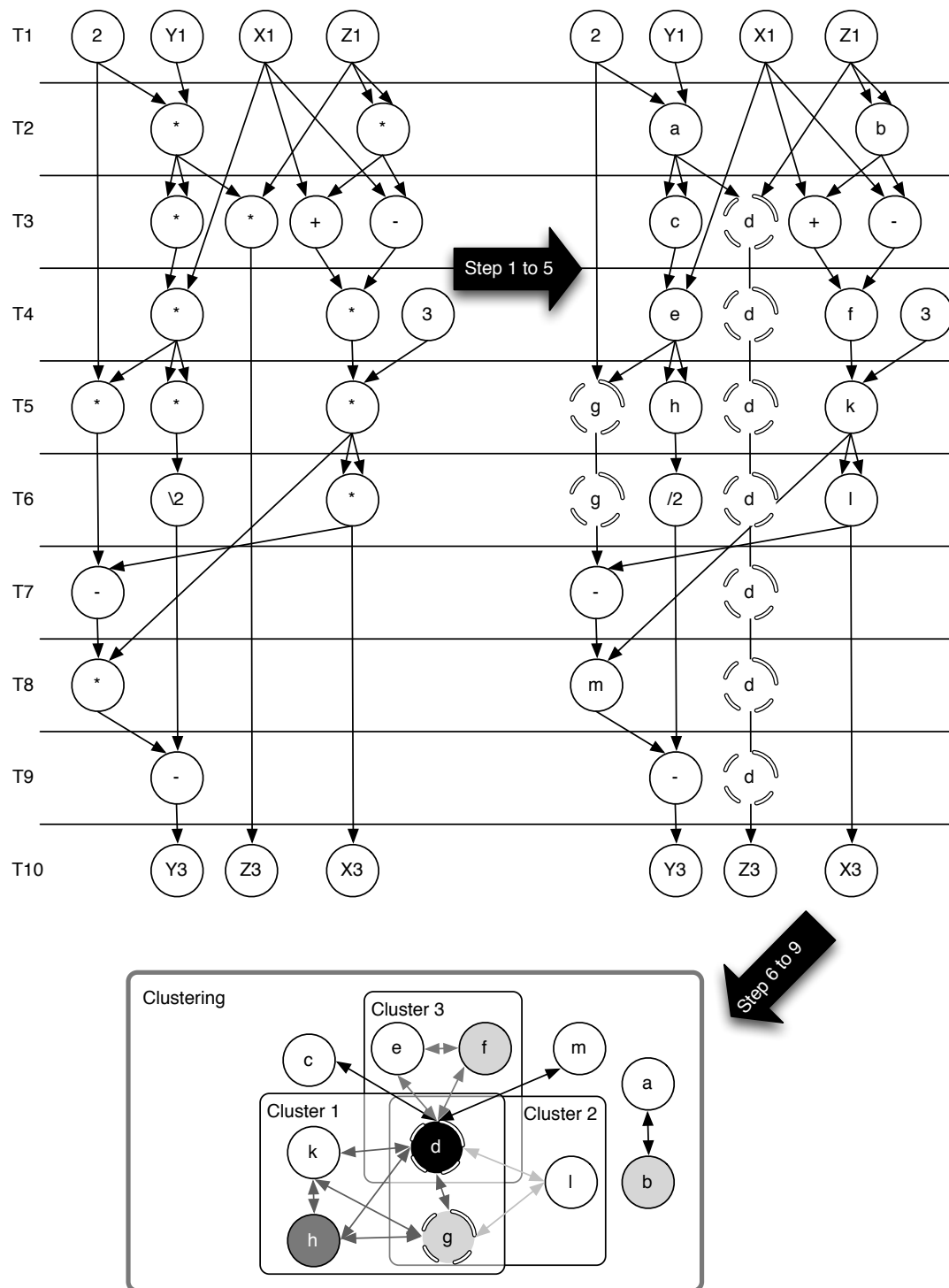


Figure 3.10: Visualisation of Algorithm 5 for a point doubling algorithm for ECC

degree of in parallel executable operational tasks is given in Algorithm 5. This algorithm is more flexible in rescheduling the DFG to a certain kind of degree and it can handle the slack of the focused processing unit operation op_{PU} due to different scheduling schemes. The calculation of the slack of a DFG is done by exploiting the scheduling algorithms *As Soon As Possible* (ASAP) and *As Late As Possible* (ALAP), both algorithms are given in Appendix A.2. In case of Algorithm 5 only the slack of the focused processing unit operation op_{PU} is calculated instead of every operation in the algorithm. Therefore, only the execution positions of the operational task types of op_{PU} are changed in the data flow. The calculated slack of op_{PU} is then used to identify the possible positions of conducting an op_{PU} task and to evaluate how it affects the amount of in parallel executable tasks of op_{PU} of the DFG in general. Thus, a rescheduling of the op_{PU} tasks without violating the complete execution duration can be handled by the algorithm in order to identify the in parallel executable op_{PU} . Based on the information of the maximum number of in parallel executable op_{PU} tasks the required number of processing units can be derived to a certain degree with regard to a flexible rescheduling of op_{PU} . In other words, Algorithm 5 describes which op_{PU} of the superior cryptographic algorithm has to be randomly concurrent executed of the processing unit in order to manipulate the power consumption signature most effectively.

A visualized procedure of the algorithm is depicted in Fig. 3.10. On the left side of the figure is the original data flow depicted. After applying the first five steps of Algorithm 5 to this data flow the result of the transformation, as visualized on the right side of the Fig. 3.10, is a new data flow graph with additional nodes. The different multiplication tasks, the focusing operation to be virtualized, are renamed with lower case letters for a clear identification of the different tasks. All multiplication tasks by a certain slack value ($op_{PU,i} \in V$ slack $_{PU,i}$) are marked with a dashed circle. The dashed marked tasks with a certain slack value are placed in all possible (according) time slots, in which they may appear due to different possible schedules of these tasks. At the bottom of the figure, the result of the transformation steps 6 to 9 of the Algorithm 5 based on the data flow graph on the right side is depicted. The different multiplication tasks are clustered into several groups according to their independency of each other. In this example, the task marked with d should be virtualized because it is the one most independent of all other tasks.

The second question is related to the need of a flexible architecture in order to bring the dynamic binding concept to practice. Therefore, the implemented architecture has to be able to reroute the data transfer between different types of processing units without compromising the overall data flow of the superior implemented algorithm. The concept of virtualization seems to be a sophisticated answer to the need of a flexible interconnection. The virtualization of the selected

types of processing units will be discussed in the next section due to its strong relation to the implementation of the dynamic binding.

3.3.3 Randomized Concurrency by Virtualization

In this subsection the concept and the realization of hardware virtualization on a FPGA platform is discussed. The application fields for the need of hardware virtualization is very broad, beginning from server clusters over operation systems on personal computers up to embedded devices. Also multiple purposes exist to conduct hardware virtualization, such as reliability (e.g.,[Yeh03]), integrity and security (e.g.,[SE08]), or resource sharing (e.g.,[BSCH11]). The focus of applying and utilizing virtualization in this case is on flexibility and dynamic resource binding for a cryptographic algorithm implementation. The benefit of such an architecture is an improved side-channel resistance by applying the principle of mutating data paths jointly with the action of a dynamic binding.

Virtualization Hardware virtualization is a method to abstract and decouple the executing hardware platform and the conducting algorithms. The proposed hardware virtualization of Stöttinger, et al. in [SBH10] uses virtualized hardware components on the FPGA platform and is used as the basis for the here described approach. A virtualized hardware component is shared by different tasks of a subroutine or algorithm, which is addressed individually by the utilizing task. The tasks of the subroutine or algorithm occupy the hardware resource without knowledge which units are shared with another set of tasks of another subroutine or algorithm. Therefore, virtualization may remove the binding between the basic processing operation and the control flow of the implemented algorithm.

Nevertheless, the tasks managed by the virtualization are transparently executed on the underlying corresponding processing units. Due to the transparent handling of the virtualization technique, it is also possible to share the same type of operational tasks between multiple corresponding processing units. So, the units may perform a set of identical and data-independent types of operational tasks in parallel, even if the original control flow of the algorithm is not intended to proceed in that manner. The property of sharing processing units between corresponding operational tasks and to share tasks between corresponding processing units qualifies the virtualization technique to implement the dynamic binding principle in practice. In conjunction with the method of online allocation, the sharing of identical operational tasks can be concurrently processed on the corresponding processing unit type with various kinds of implementations.

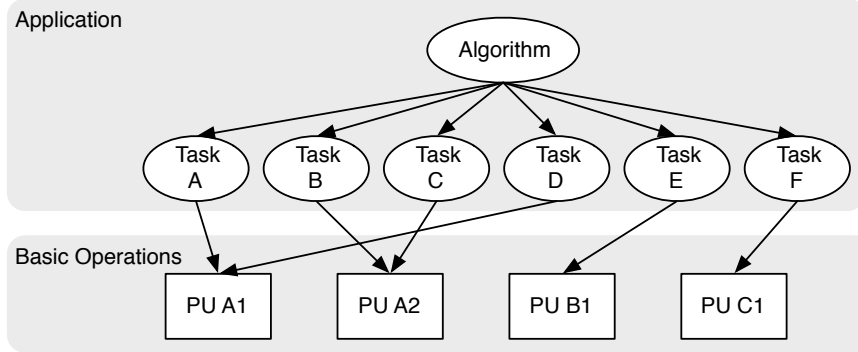
By utilizing the virtualization concept the cryptographic algorithm can be executed concurrently. The maximum degree of parallelism is given by the amount of utilized processing units and by χ obtained from Algorithm 5. Thus, the corre-

sponding $op_{PU,i}$ of the differently colored vertex of $DFG(V,E)$ can be virtualized to change the degree of parallelism. By virtualizing one of these processing unit operational tasks, the degree of parallelism is controllable by the virtualization scheme and has a strong impact on the power consumption signature of the complete cryptographic algorithm. In addition, the virtualization concept seems to be a suitable interface to combine the various proposed methods of manipulating $\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\} \leftarrow \mathfrak{dp}(\text{processing unit}, \text{concurrency}, \text{task})$, cf. Fig. 3.3.

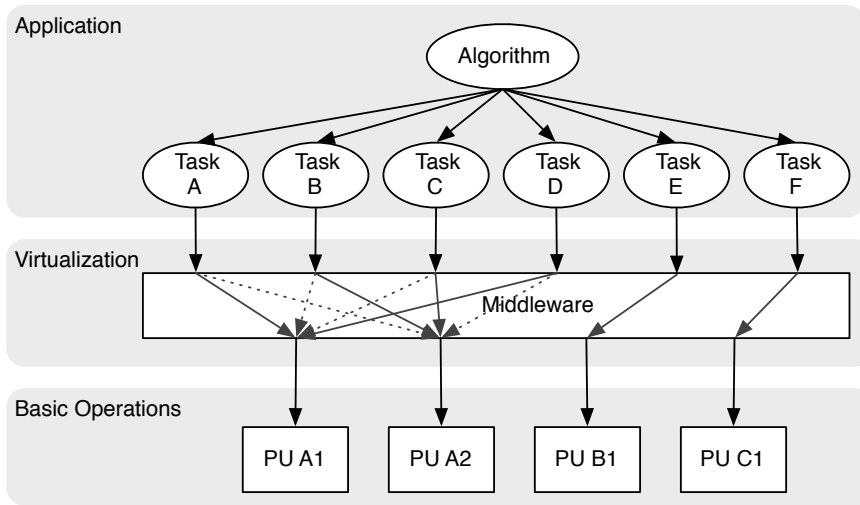
The following objectives are achievable by utilizing the concept of virtualizing processing units:

- *Online allocation during runtime*
Identifying corresponding processing units for the operational execution of the task.
- *Dynamic binding during runtime*
Mapping the (virtualized) operational tasks to appropriate processing units.
- *Flexible rescheduling of the control flow*
Adapting the control flow of the superior algorithm during and after the virtualization phase without changing the original control flow sequence.

Middleware concept The application of virtualization between the execution processing units and the sequence of tasks based on the data flow of the implemented algorithm can be established by an intermediate layer. This intermediate layer has the same functionalities as a so-called middleware, which is a commonly used method in the distributed computing area, e.g., [Ber96]. In distributed computing the middleware concept is used mainly for protocol specific tasks, like controlled data transfers and task assignments between different tasks and processing units, cf. [TS01]. Thus, the middleware principle may be used as an executive additional layer to apply the concept of virtualization for the dynamic binding approach. As depicted in Fig. 3.11, the middleware layer has an administrative task to organize the binding between the processing tasks and the executing resources. Usually this administrative process is done by an arbiter unit, which adapts the processing tasks. The basic functionality of an arbiter inside the middleware architecture is exemplarily described in Algorithm 6. First, all in parallel executable tasks T of the current schedule S_{cur} are identified.



(a) Static binding between tasks and processing units



(b) Virtualizable binding between tasks and processing units

Figure 3.11: Middleware concept for dynamic binding

After that the binding of all the unprocessed tasks is checked and sorted into two groups. One group binds all the tasks labeled with T_h . These tasks are assigned to processing units R_h , which are available at the required moment in time and can be executed according to the initial binding from the original scheduling. The other group lists all tasks, label with T_f , which cannot be executed due to their original binding, because the processing unit is not available.

In case the second group is empty, the original schedule for the current time instant S_{cur} can be executed. In the other case a new schedule for the tasks of both groups is required. Hence, the tasks T_h of the previous, not actual schedule S_{cur} will be updated to the schedule S_{aux1} . At the next step of the algorithm a new schedule (S_{aux2}) is generated based on the tasks T_f and the resources of the group R_h . After both new schedules, S_{aux1} and S_{aux2} , are established, they are executed sequentially in order to prevent execution errors due to data

Algorithm 6 Virtualization by middleware concept [SBH10]

Require: (*current schedule*)

```

 $n, m := 0$ 
 $S_{cur} \leftarrow \text{current schedule}$ 
 $t \leftarrow \text{getNumOfAllTasksOfCurrentSchedule}(S_{cur})$ 
 $T[t] \leftarrow \text{getAllTasksOfCurrentSchedule}(S_{cur})$ 
for  $i = 0$  to  $t$  do
  if ( $\neg(\text{checkAvailabilityOfResourceOfTask}(T[i]))$ ) then
     $R_f[n] \leftarrow \text{getResourceOfTask}(T[i])$ 
     $T_f[n] \leftarrow (T[i])$ 
     $n++$ 
  else
     $R_h[m] \leftarrow \text{getResourceOfTask}(T[i])$ 
     $T_h[m] \leftarrow (T[i])$ 
     $m++$ 
if ( $\text{isEmpty}(R_f[])$ ) then
   $\text{ExecuteTasksOfSchedule}(S_{cur})$ 
  return
else
   $S_{aux1} \leftarrow \text{RescheduleTasks}(S_{cur}, T_h[m])$ 
   $S_{aux2} \leftarrow \text{makeScheduleOf}(R_h[m], T_f[n])$ 
  while ( $\text{ExecuteTasksOfSchedule}(S_{aux2})$ ) do
     $\text{stallOtherTasks}()$ 
   $\text{ExecuteTasksOfSchedule}(S_{aux1})$ 
return

```

dependencies between the schedules. The schedule S_{aux2} has a higher priority in this scheme and, thus, is conducted before the S_{aux1} . This prioritization is done due to the fact the S_{aux1} is a subset of the original schedule, which requires no scheduled changes. After all tasks of schedule S_{aux1} are processed, the algorithm starts all over until all tasks are processed.

Virtualization by middleware An architecture to perform hardware component virtualization requires more functionalities than a normal data transfer and an initialization to trigger a processing unit in order to let it execute the previously provided data. The before mentioned administrative functionality requests more than the functionality of a normal data bus architecture. The middleware layer has to control the data flow, like a data bus architecture, and to perform a control flow manipulation by means of an arbiter algorithm, cf. Algorithm 6.

In the area of computer system architecture several methods exist to extend the normal data bus data routing and data transfer functionality. The basic methods to apply the middleware to a computer system architecture are (1) an arbiter controlling a switching network for data and control signals or (2) an arbiter modifying the *Very Long Instruction Word* (VLIW) commands of this flexible and in parallel working architecture. In both cases the arbiter algorithm is able to inject a new sequence of control steps and data streams in order to the corresponding processing units to execute a new alternative schedule of tasks. Both realizations have specific advantages and disadvantages:

1. The advantages of the data and control bus extended arbiter is that it is easy to be integrated in existing non-virtualizing bus architectures of an implemented cryptographic algorithm. On the downside, it will slow down the throughput of the data bus system due to the switching network. Also, the correct processing of the data strongly depends on the correct rescheduling and monitoring of the tasks by the additional arbiter unit.
2. The VLIW approach has the advantage that the virtualization is done only by manipulating the VLIW instruction sequence, which will not affect the throughput properties of the non-virtualizable design as much as the switching network approach. But the programming of the VLIW virtualization rescheduling sequences is more challenging than for the switching network.

Example of a middleware architecture for ECC In [SBH10] an implementation of an elliptic curve cryptography (ECC) based algorithm was extended by a middleware architecture in order to be able to virtualize the concurrently used multiplier units. The cryptographic algorithm operates on an elliptic curve represented in $\text{GF}(2^n)$. The original motivation of the authors was to propose a virtualizable cryptographic public key scheme, which is tolerant against faults by utilizing virtualization and still does not compromise the user storage of the virtualizing unit at runtime.

As depicted in Fig. 3.12 the middleware architecture is realized by an arbiter unit and a switching interconnection network for both control signals and data transfer. The arbiter unit exploits the administrative process described in Algorithm 6 to virtualize up to two of the three multiplication units by one unit. The virtualization CBus is used to reroute the control signals of the original instruction word to the multiplication unit, which virtualizes the inaccessible unit, while the virtualization DBus reroutes the data transfer to the executing unit. This hardware middleware approach for virtualizing inaccessible multiplication units can also be used to control the degree of in-parallel executing units in order to manipulate the overall power consumption signature.

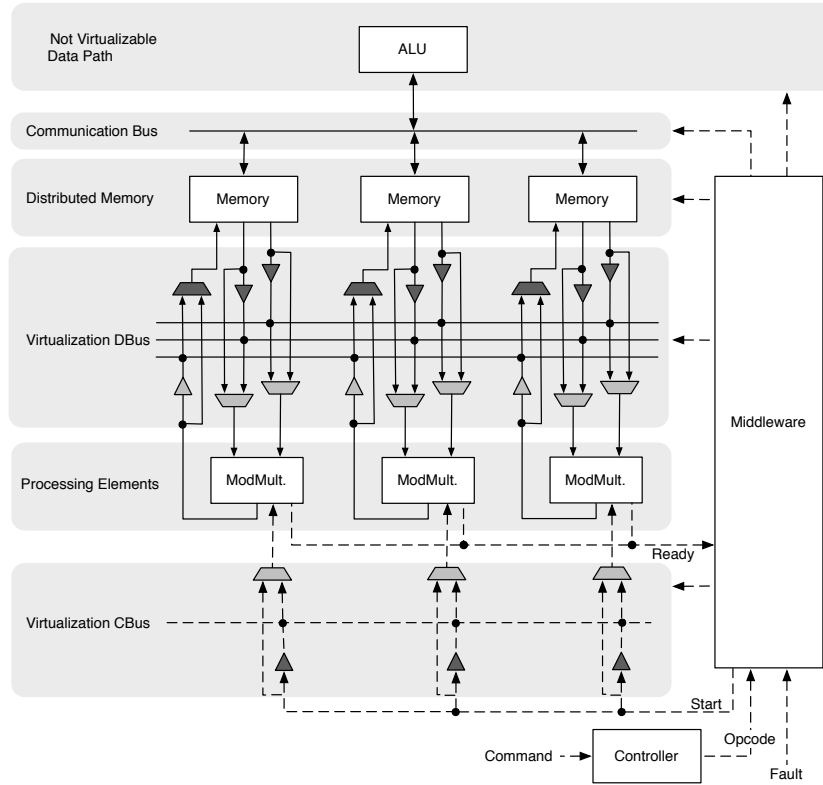


Figure 3.12: Virtualizable ECC architecture [SBH10]

3.4 Flexible Scheduling: Progression of Tasks

In this Subsect. the last design property, *progression of tasks* ($\mathfrak{dp}(\text{tasks})$), is discussed. This design property, as well as the previous ones, modifies the characteristic power consumption of the implemented cryptographic algorithm. The proposed method *flexible scheduling* is used to change the execution behavior of the cryptographic algorithm and spans the design exploration space by the third variable, cf. Fig. 3.2. This method influences the physical design behavior of the implementation by manipulating the scheduling of the tasks during runtime. The impact on the power consumption signature of the design will be detailed in this subsection, as well as the technical realization of the approach in general.

General considerations In the area of CAD, the scheduling is done during the design phase and is fixed afterwards, so that during the operation mode the progression of operations is stringently repeated for each time the crypto-

graphic operation is performed. The fixed schedule of the implemented algorithm provides the sequence of performed tasks, which includes operations to be performed. As mentioned before, after the binding is setup in the design phase in the area of CAD, the scheduling for the algorithm execution is done, thus there is a strong dependency between the scheduling and the binding.

A well partitioned control flow of the algorithm allows a feasible handling of the binding and scheduling of the resources very flexible, considering their dependency. Orientating on the control flow concepts in the area of compilers, e.g., the LLVM compiler [LLV], the control flow path consists of certain basic elements: A basic block element, a control element to fork the control flow, and a control element to join the control flow. The basic block in the control flow includes a certain amount of tasks or operations and can be abstracted as a routine in an algorithm. Routines may appear several times in the algorithm, but they only consist of a progression of tasks performed on basic operations. The flexible scheduling method is applied to the relationship and composition of such routines in the algorithm, while the dynamic binding focuses on the basic operations in a routine of the algorithm. The execution length of a routine is, of course, influenced by the degree of in parallel processed basic operations of the processing unit type within the routine. The varying execution length of the routine can be handled by the previously introduced virtualization concept of the middleware approach. Therefore, the two methods dynamic allocation and flexible scheduling, are changing the design properties of the implementing algorithm on two different hierarchical levels in conjunction with the virtualization concept, which alleviate their common application.

3.4.1 Effect on Power Consumption Patterns

The impact on the power consumption signature due to the design property $\mathfrak{dp}(\text{tasks})$ is comparable with the commonly used method of shuffling independent operations in a cryptographic algorithm, if the granularity of processed operations per clock cycle does not change. Therefore, the flexible scheduling recomposes the routine in a manner that only the sequence of data-independent basic operations within the routine is manipulated. This effect also holds for the shuffling of the sequence of complete routines swapped in the algorithm, if this does not compromise original results of the cryptographic algorithm in terms of correct calculation. The increased approximation of the effort for attacking a system, which shuffles its data-independent operations, is reported in [MPO07]. Based on this approximation for a design with shuffling⁵ the effort can be ex-

⁵The operation shuffling is equally distributed with fixed amount of operations per clock cycle

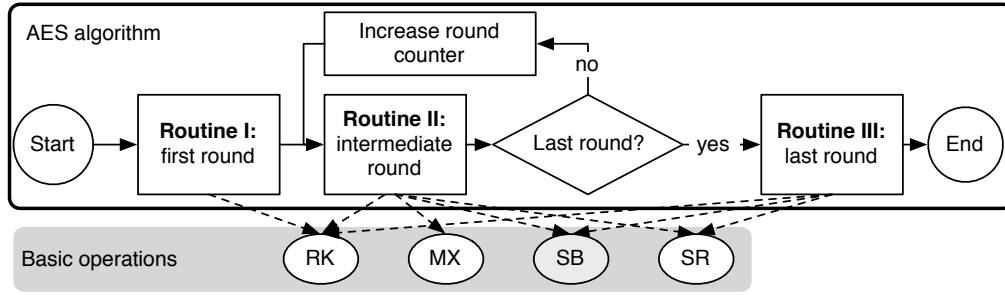
pressed in n_{measure} times additional number of power traces, cf. Eq. (3.12).

$$n_{\text{measure}} = (\text{number of shuffling operations})^2 \quad (3.12)$$

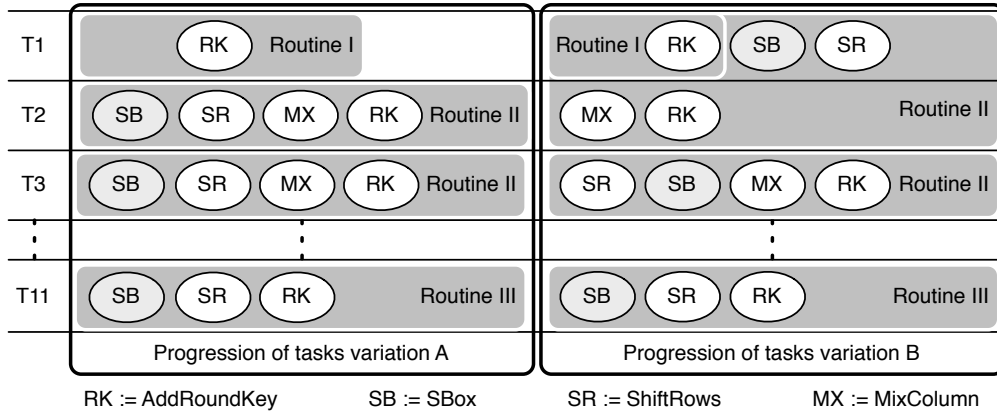
Increasing the effort can be achieved by chaining the amount of atomic operations in a routine, in terms of numbers of operations that are executed within one clock cycle. This approach is similar to the mentioned countermeasure *Temporal Jitter* in Subsect. 3.1. The varying number of performed operations within one clock cycle influences the variance of the power consumption within the clock cycle and thus decreases the signal-to-noise ratio. An approximation for the influence of a changed signal-to-noise ratio is given in [MPO07], but this approximation bases on the utilization of CPA.

Another improvement to increase the effort of extracting exploitable information by terms of a flexible scheduling is to recompose the routines of the algorithm with different basic operations. The advantage of recomposing the routines with different progressions of basic operation tasks compared to shuffling the execution order of the basic operations is that the intermediate values of the routine are changed, instead of being misaligned in the time domain. Therefore, the attack gets more complex because more possible intermediate values have to be considered for the estimation of the power consumption.

Comparing the previous methods of online allocation and dynamic binding for mutating $\mathbf{dp}(\text{processing unit}, \text{concurrency})$ at first glance the flexible scheduling method seems to be the same method on the software level. At a closer look, the first two methods work on the same type of basic operation in the algorithm in order to mutate $\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\}$, while the manipulation of $\mathbf{dp}(\text{task})$ works on different types of basic operations and thus different types of hardware components. By this the variations on the during runtime reconfiguring processing unit with the most significant power consumption profile is enveloped in the randomization process caused by the method of flexible scheduling. Therefore, a pattern detection via a SPA attack on specific $\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\}$ by exploiting the connection of the corresponding $\mathbf{dp}(\text{processing unit}, \text{concurrency}, \cdot)$ to the power consumption signatures gets more difficult. Otherwise, if an SPA attack could gain enough information to confidently distinguish the different design properties an attack in two phases is possible, similar to a second order attack. In the first phase, the traces are clustered according to their distinguishable SPA signature and then, in the second phase, each cluster can be attacked individually. Thus, the third $\mathbf{dp}(\cdot, \cdot, \text{task})$ extends the possibilities of causing a specific $\{\sigma_{\mathcal{P}_{\mathcal{L}_t}}^2, \mu_{\mathcal{P}_{\mathcal{L}_t}}\}$ as well as it is expanding the design exploration space.



(a) Control flow of the AES algorithm



(b) Example of rearranged basic operations in a routine

Figure 3.13: Subroutine rearrangement by routine (re)shaping

3.4.2 Partitioning the Algorithm

In this subsection the implementation of partitioning is discussed based on the investigated algorithm. Thereby it enables one to rearrange routines of the algorithm or to completely recompose the routines. The flexible scheduling exploits the modularized structure of cryptographic algorithms. The intuitive procedure to capsule frequently used functionalities or procedures of an algorithm into routines supports the partitioning process in order to establish a flexible scheduling. The interface⁶ of the routines to the algorithm has to be identical for each different type of routine implementation. In contrast to the interface, the computational sequence and intermediate values of the routine itself has to vary. For each of the randomization techniques of the flexible scheduling method an example will be provided in the following.

Rearrangement of the basic operations in a routine Most cryptographic algorithms are constructed on basic operations or exploit other cryptographic

⁶Routine parameter provided from the algorithm and the returned result of the routine

functions, which are composed of basic operations. For instance most block cipher algorithms, which are members of the symmetric-key cryptographic algorithms, e.g., AES or PRESENT, are round oriented algorithms. As depicted for the example of an AES data path in Fig. 3.13(a), each different round function of the AES can be implemented in its own routine, which utilizes the operations out of a common set of basic operations. Each of the three routines can be manipulated in their physical behavior by means of different executed basic operations in one clock cycle, without changing neither the degree of parallel processed bits, nor the hardware implementation of the basic operations.

In the example provided in Fig. 3.13, the basic operation of the SBox is handled as the mutable processing unit, which can be reallocated online while the device is active. In addition, the degree of in parallel processing SBoxes can be manipulated by the dynamic binding method. As the rearrangement example in Fig. 3.13(b) demonstrates the first round of the AES, which is one of the two common targeting scenarios of a power analysis attack, can be merged with some basic operations of the second routine. Therefore, the number of operations executed within one clock cycle is changed from the usual scenario on the left side of Fig. 3.13(b) to the rearranged routine, on the right side of Fig. 3.13(b). Of course, it is also possible to change the amount of basic operations of *routine II* for each clock cycle. The power consumption signature is disturbed by a sum of the Gaussian distribution that is caused by the different active operations during one clock cycle over the observed set of experiments with varying operational activity. Thus, the resulting complex joint distribution differs from the distribution in Eq. 3.11 and it includes the dynamic binding activity as well as the changes of the processing unit design due to the online allocation.

Recomposition of the basic operations in a routine In case of a more complex algorithm, with multiple hierarchical arithmetic layers, for instance like in elliptic curve cryptography, the routines of the higher hierarchical levels can be composed from various basic operations. The method of recomposing the routine is demonstrated on the example of the point doubling operation ($[2]P$) for an elliptic curve E in $GF(p)$. The projective representations are exploited to compose two different routines for the point doubling operation.

As depicted by the abstractly denoted Algorithm 7 and Algorithm 8, the intermediate values of both routines differ due to their own procedures of computation. Of course, due to the different point representations, the results of the two routines are not the same as well, but they can be transformed to a commonly affine representation for exchanging the values between the routines.

Algorithm 7 Point doubling projective, $[2]P = (X_3 : Y_3 : Z_3)$ based on algorithm from [CF05]

Require: A point $(X_1 : Y_1 : Z_1)$ on curve E

Ensure: An affine point (x,y) on E equals $(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1})$ with $Z_1 \neq 0$

- 1: $A = a_4Z_1^2 + 3X_1^2$
 - 2: $B = Y_1Z_1$
 - 3: $C = X_1Y_1B$
 - 4: $D = A^2 - 8C$
 - 5: $X_3 = 2BD$
 - 6: $Y_3 = A(4C - D) - 8Y_1^2B^2$
 - 7: $Z_3 = 8B^3$
 - 8: **return** $(X_3 : Y_3 : Z_3)$
-

Algorithm 8 Point doubling Jacobian, $[2]P = (X_3 : Y_3 : Z_3)$ based on algorithm from [CF05]

Require: A point $(X_1 : Y_1 : Z_1)$ on curve E

Ensure: An affine point (x,y) on E equals $(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1})$ with $Z_1 \neq 0$

- 1: $A = 4X_1Y_1^2$
 - 2: $B = 3X_1^2 + a_4Z_1^4$
 - 3: $C = -2A$
 - 4: $D = 8Y_1^4$
 - 5: $X_3 = C + B^2$
 - 6: $Y_3 = D + B(A - X_3)$
 - 7: $Z_3 = 2Y_1Z_1$
 - 8: **return** $(X_3 : Y_3 : Z_3)$
-

Two different routines to perform a point doubling operation ($[2]P$) on a curve E, described by the affine form $y^2 = x^3 + a_4x + a_6$

One important issue to notice is that the composition variants of the routine should not be executable in one clock cycle, especially if the procedure of computation is different, but lead to the same final value. Multiple clock cycles should be used and, if necessary, an additional hardened version to transform the result into a common representation in order to hinder an exploitation of register transition of this final value. The recomposition of a routine has to lead to different computations and thus, to different intermediate values stored in the shared registers of the compositions of the routine. In addition, the composition schemes of the routine should exhaust the same amount of clock cycles in order to prevent an identification of the schedules by a timing analysis. Utilizing the flexible scheduling method to recompose a routine is more complicated to achieve than merging different operational tasks of different routines into one clock cycle, but it has a bigger impact on the power consumption of the implementation.

3.4.3 Randomized Micro-Program Selection

Due to the similarity of flexible scheduling to dynamic binding as well as to online allocation, its implementation orientates on the previously discussed techniques of the other two proposed methods. Therefore, a data path implementation with a flexible scheduling ability requires the following two main characteristics:

1. A transparent interface scheme to handle the control flow from the algorithm to the various implementations of the routines as well as between the different routine types.
2. A flexible data access to the requested parameters each routine needs in order to perform its computational procedures at any state of the algorithm.

The technique of virtualization is well-suited to meet these requirements. A hardware/software-codesign (HW/SW-codesign) structured architecture therefore seems the most effective solution for the implementation of a data path that supports flexible scheduling as well as the other two proposed methods. Therefore, the theoretical, into routines partitioned algorithm is then implemented into different micro-code programs. A micro program has to be implemented for each routine and for each computational procedure of the different routine types. In addition, the main control flow of the implementing algorithm has to be provided.

When utilizing the previously introduced virtualization in conjunction with a VLIW architecture each micro program can be easily executed within the control flow of the algorithm. In the HW/SW-codesign architecture the micro programs can be executed by the main control flow via their start address. The control flow is then handled by the middleware of the virtualization scheme. An important design criterion for establishing the memory map of the programs is to consider the start address of every routine. In order to prevent power analysis attacks on the routine call or, to be more specific, on the kind of implementation of a routine, the addresses of each routine entity should have the same Hamming weight. Figure 3.14 depicts an example of how to position the micro programs in the memory to prevent a side-channel leakage by the Hamming weight of their routine addresses. In this example the addressing of the implementation versions of the routines can be done by a random generator in order to randomly address the one-hot coded implementation version of a routine. As visualized in Fig. 3.14 the four most significant bits are used to randomly select an implementation version, followed by the next four bits to select the routine type. The other bits down to the LSB are used to address the basic operations within each implementation of each routine.

A shared memory architecture is one possibility to achieve the second request for the flexible scheduling method in practice. This memory concept provides multiple access to differently stored values and thus it can store all necessary values for the cryptographic algorithm as well as for the different routine implementations. Thereby, the processing units and hardware components can access the values following the schedule of the randomly selected routines of the cryptographic algorithm. With this proposed architecture the $\mathbf{dp}(\text{processing unit, concurrency, task})$ of the data path can be adjusted during

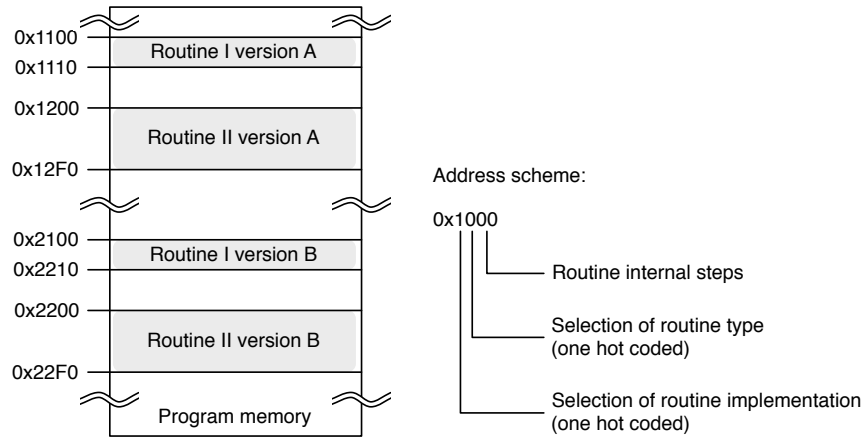


Figure 3.14: Addressing scheme of the micro programs

runtime by using flexible scheduling in conjunction with dynamic binding as well as online allocation. They will not interfere with each other in a functional manner, but in a way, where the power consumption characteristics will be blurred.

3.5 Design Flow of Mutating Data Paths

In this Subsect. the overall design flow for creating mutating data path structures is detailed. The design flow utilizes all the previously described proposed methods to manipulate the \mathfrak{dp} (processing unit, concurrency, task) during runtime. The described procedures are to be performed step by step and are not yet completely automated. Figure 3.15 visualizes the procedure of the design flow of getting a design with a mutating data path starting from a given cryptographic algorithm, for instance a block cipher.

At first a partitioning of the hardware and software for the implementation of the given algorithm has to be made. Usually, all the procedures dominating the control flow are grouped into software while the basic operations to be performed are grouped into hardware. Procedures, which are composed of basic operations, can also be mapped to the software partition in order to reuse hardware resources on the data path.

The information clustered in the software partition are then used to establish different schedules by utilizing the flexible scheduling method. As mentioned before the flexible scheduling method can either be used to recomposition routines or to rearrange the procedures within the routines, cf. Sect. 3.4. At first for both methods the routines are identified in the control flow of the software partition. After that the different schedules for both methods are generated, if possible.

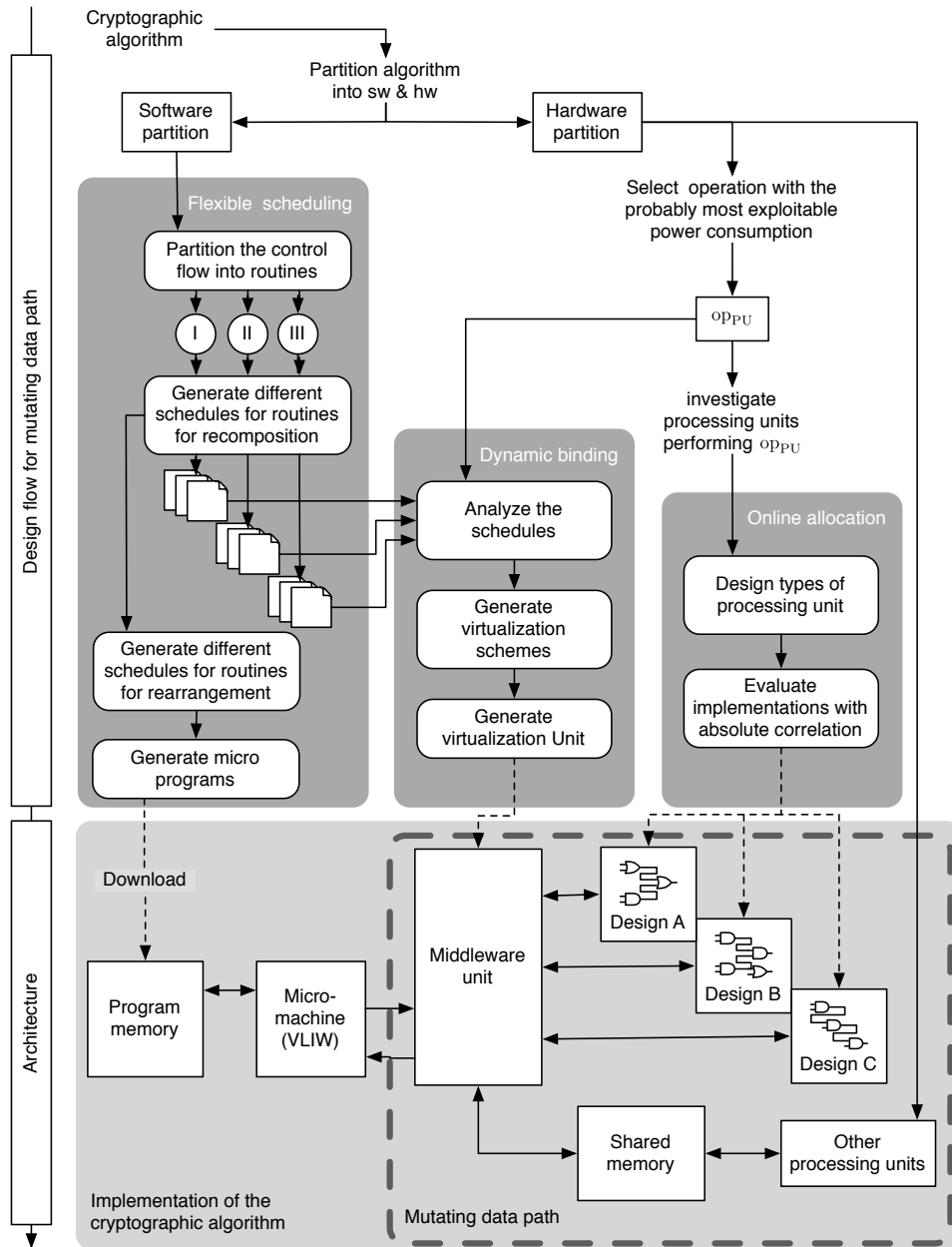


Figure 3.15: Design flow to create a mutating data path and its generic architecture

If not, only the corresponding schedule for the applicable method is generated. The scheduling of the rearrangement method is then used to generate microprograms and storing them in a sophisticated scheme in the program memory, cf. Subsect. 3.4.3. The other schedules for the recomposition are made available to the processing step utilizing the method of dynamic binding.

The information within the hardware partition is used for the previously proposed methods, i.e. online allocation and dynamic binding. At first, the basic operation conducted on a processing unit with the most exploitable power consumption op_{PU} is identified. The processing units performing the other basic operations can be implemented directly in order to be used in the data path architecture as depicted at the bottom of Fig. 3.15. The provided schedules are then analyzed in the basic design process dynamic binding in order to identify the usage of op_{PU} in the schedules. After this step, the dynamic binding procedure generates the virtualization schemes for op_{PU} according to the description in Sect. 3.3. These schemes are then utilized to construct the related virtualization unit. In parallel to the dynamic binding and flexible scheduling process the basic design process online allocation can be performed. As reported in Sect. 3.2 this basic design process method is used to identify suitable different implementations of a processing unit to hide the data-dependent power consumption. At first, different implementations of the same provided functionality of the processing unit PU are implemented for the operation op_{PU} . After that the power consumption behavior of the various implementations are compared and the most suitable ones are selected, cf. Subsect. 3.2.2.

In the end after all three design steps have been performed the components can be assembled to the mutating data path architecture with a virtualization unit and various different types of processing unit implementations. The randomization of the active processing unit is handled by the middleware (the virtualization unit) via a switching network or by exploiting partial reconfiguration for Xilinx FPGAs, cf. Sect. 3.2.3.

Analysis Framework

Contents

4.1	General Evaluation of Side-Channel Resistance	68
4.1.1	Evaluation Procedure	72
4.1.2	Evaluation Metrics	75
4.2	CSA-Based Evaluation Metric	79
4.2.1	Constructive Side-Channel Analysis	80
4.2.2	Signal-To-Noise Based Evaluation Metric	86
4.3	Acquisition Setup	90

For side-channel resistance evaluation it is crucial not only to detect side-channel flaws in the circuit, it is even more important to characterize the unintended information leakage. A quantitative characterization of both the exploitable leakage and the corresponding circuit section are used to pinpoint the origin of the side-channel leakage.

In this chapter the elaborated framework of this thesis for the side-channel evaluation against power attacks is introduced. First, a general overview of the state of the art in side-channel evaluation is given before the analysis framework of this thesis is described. The generic framework for any side-channel evaluation consist of an acquisition setup and related side-channel analysis tools. The acquisition setup is used to capture the measurable leakage from the device during its normal operation mode, while the analysis tools are intended to characterize the thread of power analysis attacks of the analyzing implementation. A direct comparison between the different implementation approaches is therefore applicable in order to improve the side-channel resistance of the circuit. Thus, the dedicated analysis tools detailed in the sequel gain the insight of the here proposed hardened design techniques in terms of their power analysis attacks resistance.

4.1 General Evaluation of Side-Channel Resistance

The basic goal of a side-channel resistance evaluation is to gain insight of a circuit and to identify exploitable leakages that compromise the implemented algorithm's secret. In case of detecting an exploitable leakage the circuit has to be evaluated how feasible it is to conduct an attack on the implementation and to which degree the secret is being revealed. In 2009 Standaert et al. followed these general concepts and introduced a generic side-channel analysis framework, cf. [SMY09].

Nevertheless, it seems that there is no common understanding and usage in the community, of the term *leakage* and no standardized mathematical identifier or formal description of the *transfer-function* that shall characterize the leakage based on the algorithm's parameters. There are multiple and very coarse descriptions of the term leakage. Also the understanding of a leakage model seems to vary among different authors or is not stated precisely yet, cf. [MOS09, WO11a]. For instance, in some cases there exists an understanding of leakage models, which represents more or less a functional description of mapping the internal behavior of the circuit to the emitted leakage [MPO07, EG10]. In other cases the mapping is done by a so called leakage function, which relies on a physical understanding of the internal circuit behavior, cf. [MR04, SMY09, KSS10, SVCO⁺10]. The basic concepts in [SMY09] seem to be solid at a first glance, but not all definitions are free of term collisions with the area of circuit design, for instance the definition of an implementation:

"An implementation is the combination of a cryptographic device and a leakage function." [SMY09]

This definition may hold for the manufactured silicon at the end of the design process, if each implementation of the algorithm is hosted in a new device. It definitely holds for implementing a cryptographic algorithm in software on a CPU or on an embedded device with an fixed hardware architecture, for instance a μC . But it does not hold in case of implementing different circuits with the same algorithmic functionality on one and the same FPGA device, which may feature considerably different non-functional properties. Thus, the definitions of [SMY09] can not be used directly in this thesis, which has a strong focus on FPGAs. So, it seems substantial to provide a foundation in this thesis for a common understanding of the basic terms in order to increase the transparency and understanding of this thesis and the proposed evaluation techniques and methods. The definitions for the common basis of terms and transformations given here aim at a general understanding of side-channel analysis and thus are not

restricted to the area of power analysis attacks only. Nevertheless, explanatory examples of these terms are provided on instances of the areas of power analysis only as they are in the focus of this thesis.

Definition 1 provides a general understanding of a leakage \mathcal{L}_t that is emitted from the implemented circuit inside an active device. This unintended source of information may be compromising the security of the implemented algorithm. So, it should be clarified that leakage \mathcal{L}_t is an observable physical quantity in the environment of the device. Therefore, the leakage \mathcal{L}_t features both a real number value and an appropriate physical unit.

Definition 1

A leakage \mathcal{L}_t is an unintended information source about the device's circuit internal secrets, which is observable by phenomena in the physical domain at certain points in time. $\mathcal{L}_t \in \mathbb{R}$ and has the unit of the corresponding physical domain.

The second definition, Def. 2, is crucial for an understanding of the origin of the unintended secret information broadcast. By this definition a leakage model $\mathcal{L}_{\mathcal{M}_t}$ is a deterministic relation between the internal activity of the circuit and behavior and the leakage \mathcal{L}_t . Furthermore, it quantifies that the emitted leakage \mathcal{L}_t is a reaction of internally processed values and thus any leakage also depends on the algorithm's internal states or transitions. These internal states and transitions are caused by public and secret algorithm parameters and thus establish a link to the physical domain. For instance, the Hamming distance model or Hamming weight model are commonly used leakage models in power analysis attacks for CMOS based designs. These two models exploit in contrast to the static case the higher dynamic power consumption in CMOS technology-based circuits caused by their switching activity.

Definition 2

*A leakage model $\mathcal{L}_{\mathcal{M}_t}$ gives a **causal** relationship between a reasonably measure-able leakage \mathcal{L}_t emitted from the implementation at the time instant t and an estimated leakage $\tilde{\mathcal{L}}_t$ based on the processed data on the device at t .*

The leakage model $\mathcal{L}_{\mathcal{M}_t}$ utilizes one or multiple leakage functions $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ to characterize the circuit's internal behavior that causes the measurable leakage \mathcal{L}_t . It is essential for deriving the leakage function $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ to exploit both the

knowledge on the implemented algorithmic procedure and on the technology of the platform. Please note that a leakage model $\mathcal{L}_{\mathcal{M}_t}$ can utilize different leakage functions $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ to characterize the same leakage \mathcal{L}_t . In general, the input parameter \mathbf{x} denotes one or multiple public or accessible algorithm parameters, which may be exploited by its or their functional relationship to a secret algorithm parameter (\mathbf{y}) within an algorithm's intermediate operation. Thus, the $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ can be exploited to reconstruct intermediate values of the algorithm, which cause an exploitable leakage and therefore compromises the algorithm's secret. The input parameters of the leakage function $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ are parameters of the implemented algorithm, like a fragment of the ciphertext or the plaintext, and a part of the secret key (subkey) in case of modeling an encryption cipher.

At least one secret parameter is needed in the function to map the information theoretical data to the physical leakage. The output of the function is the leakage \mathcal{L}_t . Definition 3 mathematically defines the term of a leakage function used in this thesis:

Definition 3

One or multiple leakage functions $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ are contained in a $\mathcal{L}_{\mathcal{M}_t}$ quantifying the relationship between the leakage \mathcal{L}_t and the estimated leakage $\tilde{\mathcal{L}}_t$. Therefore, any leakage function $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ maps the input parameters (\mathbf{x}, \mathbf{y}) with $\mathbf{x} \in \mathcal{X}^n, \mathbf{y} \in \Upsilon^m$ to the exploitable leakage \mathcal{L}_t of the device with $\mathcal{L}_t \in \mathbb{R}$. The sets \mathcal{X} and Υ and their sizes (n, m) depend on the implemented algorithm running on the device. The public or accessible algorithm parameters, like device specific input or device output parameters, are denoted by \mathbf{x} . The algorithm's secret parameter is marked with \mathbf{y} . Those two types of parameters are called algorithm specific parameters.

For instance, in case of a power analysis attack on an 8 bit xor-operation, a leakage function relying on a Hamming weight (HW) model can be expressed as:

$$\begin{aligned} \mathcal{L}_{\mathcal{F}_{t,(a \oplus b),\text{HW}}}(a, b) &= \sum_{i=1}^8 a_i \oplus b_i = \sum_{i=1}^8 (a \oplus b)_i, \\ \text{with } a &= \sum_{i=1}^8 a_i \cdot 2^i, \quad b = \sum_{i=1}^8 b_i \cdot 2^i \end{aligned} \quad (4.1)$$

Based on the definitions Def. 1 - 3 the basic tasks of a side-channel evaluation are *leakage characterization* and *design issue correction*. The following task description is given in a general case for power analysis and thus can be applied

in a straight-forward manner to any of the different power analysis methods. In addition to the above stated tasks, the area of *constructive side-channel analysis* (CSA) emphasizes the quantification of the leakage characterization due to the analyzed circuit properties, cf. [KSS10].

Leakage characterization Based on the designer’s or the evaluator’s knowledge about the implementation itself and the circuit-realization of the algorithm certain possible leakages from the security sensitive parts of the implementation are assumed. The accuracy of the evaluation (identification and characterization of the assumed leakages) strongly depends on the selected $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ and thus on $\mathcal{L}_{\mathcal{M}_t}$. The better the $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ maps the physical circuit behavior, the better the physically measured leakage from the target circuit can be characterized. In case of a power analysis the power consumption on the target device is captured multiple times in a separate acquisition process, cf. Sect. 4.3. Usually, the characterization of the leakage is only the evaluation of the attack feasibility in terms of the effort during the acquisition and the needed computational power in order to perform the attack in a feasible time interval. This characterization of the side-channel is done by one of the different evaluation metrics used in the end of the evaluation process, cf. Subsect. 4.1.2.

Design issue correction The results of the side-channel resistance evaluation should provide the designer or evaluator with an insight into the attack vulnerability of the analyzed circuits parts. Please note that the evaluation results strongly depend on the selected $\mathcal{L}_{\mathcal{M}_t}$ and $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$. Thus, not sophisticated versions of these may lead to a false confidence about the security of certain circuit components, cf. [EG10, HKSS11]. The insight into the circuit also strongly depends on the leakage functions $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ from another point of view. The result of the evaluation phase provides only the information whether the assumed leakage described by the $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ exists and how much effort (e. g., amount of captured power traces) has to be considered to prove the correctness of the hypothesis. Depending on the designer’s or evaluator’s expertise, the implementation flaws of the attackable circuit component are identified based on this binary insight and the source of side-channel leakage is estimated. In other words, the more leakage functions an evaluator used during the evaluation, the better she or he can identify the implementation flaws and try to correct these weaknesses.

CSA-based aspects The focus of CSA is to give a designer or an evaluator a more detailed feedback about the circuit issues instead of verifying a leakage assumption depending on the selected $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ in a binary way. The CSA ap-

proach can extend the basic concept of a straight forward side-channel resistance analysis, as described in the two paragraphs above. The additional gain is mainly achieved by a more complex $\mathcal{L}_{\mathcal{M}_t}$, leading into a $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ that still depends on an intuitive leakage assumption of the implementation weaknesses, but also may include a circuit specific expertise of the designer. Thus, the granularity of the model can be varied depending on the implementation knowledge level and the focused circuit components properties. Due to the profiling-based approach a verification of the accuracy of the selected $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ to counteract false confidence about the security may also be supported. The ability of checking the leakage model by the CSA concepts will be more detailed in the Subsect. 4.2. The extended $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ and its suitability for the side-channel evaluation in this thesis will be discussed in this Subsection too.

4.1.1 Evaluation Procedure

The evaluation procedure of power analysis may be divided into four different phases. All these phases are detailed in the following paragraphs and the algorithmic description at the end clarify their positions in the general workflow of a power analysis evaluation procedure. All tasks are described independently from the different power analysis attacks, cf. Subsect. 2.2.1.2, in order to depict the core concepts, which are applicable to leakage exploitation in the power consumption domain. The focus of these tasks is side-channel evaluation of a data oriented leakage exploitation, like the DPA in the area of power analysis. Nevertheless, with minor adaptations the evaluation procedure works also for control flow orientated analysis¹ as a subset of the described tasks.

Leakage consideration In the beginning, depending on the insight of the algorithm implementation on the device, sophisticated assumptions about the unintended information leakages can be formulated. Thus, information leaking critical security-sensitive components can possibly be identified. For each critical component an exploitable leakage \mathcal{L}_t is assumed by the designer or evaluator due to his expertise. Based on these assumptions and the utilized technology platform an appropriate $\mathcal{L}_{\mathcal{M}_t}$ and consequently a sophisticated $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ for each \mathcal{L}_t are derived. The leakage functions $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ are later on utilized to calculate an estimated hypothetical leakage $\tilde{\mathcal{L}}_t$ caused by \mathbf{x} and related hypothetical possible values of \mathbf{y} .

Captured leakage The actual leakage of the implemented algorithm within the device under test can either be captured from an active implementation of

¹such as SPA for power analysis

a physical device or from a simulated physical model of the envisaged implementation. These captured data, so-called *traces*, are generated by varying the accessible public parameters \mathbf{x} and keeping the secret parameter \mathbf{y} static. After all the traces with one static value of \mathbf{y} are recorded, the value of \mathbf{y} is changed and the capturing process is repeated with the the same values of \mathbf{x} used with the previous \mathbf{y} value. In case of performing an evaluation with profiling-based analysis methods² traces are captured for all possible \mathbf{y} values with an identical set of \mathbf{x} . Later on these traces are used to specify the model for the power consumption by customizing the $\mathcal{L}_{\mathcal{M}_t}$ to the analyzed implementation. The varying algorithm parameter \mathbf{x} , which is fed into the device or simulation, is also stored in order to estimate the hypothetical leakage $\tilde{\mathcal{L}}_t$ for every possible value of the secret parameter by utilizing the $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$. For instance, in case of capturing the leakage from an encryption scheme, these parameters would be the plaintext and the secret key, respectively. The captured leakage can be interpreted as a random variable with an unknown distribution depending on the algorithm's public and secret parameters.

Calculate secret-based hypothetical leakage The $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ is utilized to calculate the hypothetical expected $\tilde{\mathcal{L}}_t$ for each tuple (\mathbf{x}, \mathbf{y}) based on every possible value of \mathbf{y} and the stored values of \mathbf{x} . The stored values of \mathbf{x} stem from the leakage capture procedure. In general, the hypothetical expected leakage is determined in two steps. A leakage function $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ exploits intermediate results and states of an algorithm and, thus, calculates intermediate values depending on the values of \mathbf{x} and \mathbf{y} . The afterwards performed transformation of the number values into the estimated leakage values in the physical domain is done with regard to the underlying $\mathcal{L}_{\mathcal{M}_t}$. For profiling-based evaluation methods the previously additionally captured traces for all possible values of \mathbf{y} are utilized to customize the $\mathcal{L}_{\mathcal{M}_t}$ to the analyzing implementation. In general, this customization of $\mathcal{L}_{\mathcal{M}_t}$ is done by reconstructing the probability distribution of the \mathcal{L}_t value for every \mathbf{y} by deploying $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ of the primarily $\mathcal{L}_{\mathcal{M}_t}$, cf. [CRR02].

Evaluate side-channel leakage The evaluation phase of the side-channel leakage is separated into two steps. First, it has to be verified whether there exists an exploitable leakage. Therefore, a side-channel distinguisher $\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_t)$ is used. It compares the hypothetical leakage $\tilde{\mathcal{L}}_t$ for every value of the secret algorithm parameters with the captured leakage \mathcal{L}_t taken from the device under analysis. By this procedure all the leakage assumptions and their estimated hypothetical leakage from the previous phase are verified. The goal of the first step is to get a ranking of the most likely \mathbf{y} value, which causes the most similar

²see Subsect. 2.2.1.2

Algorithm 9 Common side-channel evaluation of implementation I of algorithm A

Require: Known functional behavior of algorithm A

- 1: Consider possible emitted $\mathcal{L}_{t,I}$ of I
 - 2: Select appropriate $\mathcal{L}_{\mathcal{M}_{t,I}}$
 - 3: Formulate $\mathcal{L}_{\mathcal{F}_{t,A,I}}(\mathbf{x}, \mathbf{y})$ based on $\mathcal{L}_{\mathcal{M}_{t,I}}$
 - 4: **for** i-times select a $\mathbf{y} \in \Upsilon^m$ **do**
 - 5: **while** Performing $j \in \{1, \dots, N_1\}$ measurements the running implementation I on the device with selection of a $\mathbf{x} \in \mathcal{X}^n$ **do**
 - 6: Capture the emitted $\mathcal{L}_{t,I_{i,j}}$ and the parameter of (\mathbf{x}, \mathbf{y})
 - 7: **for** $l = 1$ **to** $\dim(\Upsilon \cdot m)$ with $\tilde{\mathbf{y}} \in \Upsilon^m$ **do**
 - 8: Estimate the expected $\tilde{\mathcal{L}}_{t,I_{i,j,l}} \xleftarrow{\text{predicted}} \mathcal{L}_{\mathcal{F}_{t,A,I}}(\mathbf{x}, \tilde{\mathbf{y}})$
 - 9: Post-processing of the captured traces to reduce noise
 - 10: Use a distinguisher $\mathcal{D}(\mathcal{L}_{t,I_{i,j}}, \tilde{\mathcal{L}}_{t,I_{i,j,l}})$ to rank all $\tilde{\mathbf{y}}$ (from *min* to *max*)
 - 11: Store ranking in res_i
 - 12: Evaluate res_i with evaluation metric
 - 13: **return** Averaged metric result over all i-test cases
-

$\tilde{\mathcal{L}}_t$ compared to the captured \mathcal{L}_t . In the second step of the evaluation phase the level of exploitation of the identified \mathcal{L}_t is determined by utilizing one or several evaluation metrics. The metrics focus on the effort to successfully exploit the side-channel leakage emitted from the analyzing implementation in order to reveal the secret parameters of the algorithm. They provide the information of how many traces have to be considered in the evaluation to guess the secret correctly, i.e., to distinguish between exploitable \mathcal{L}_t and noise.

Algorithm 9 depicts the typical workflow of a side-channel evaluation investigating an implementation I of an encryption algorithm A . The description of this generic example focusing on the area of power analysis is to depict the application in the focused side-channel area of this thesis. In case of a power analysis $\mathcal{L}_{t,I}$ is the exploitable data dependent power consumption $\mathcal{P}_{\mathcal{L}_{t,I}}$ of the device. Due to the platform dependability of $\mathcal{L}_{\mathcal{M}_{t,I}}$, it is most likely to use the switching activity based HD power model in case the implementation I bases on CMOS technology. Thus, the corresponding leakage functions $\mathcal{L}_{\mathcal{F}_{t,A,I}}(\mathbf{x}, \mathbf{y})$ are utilized to estimate the data-dependent power consumptions caused by a certain value of \mathbf{y} and multiple values of \mathbf{x} . The \mathbf{x} values have to be members of the set \mathcal{X}^n , which represents all possible values of the used public parameter in $\mathcal{L}_{\mathcal{F}_{t,A,I}}(\mathbf{x}, \mathbf{y})$.

In a power analysis evaluation the estimated leakage $\tilde{\mathcal{L}}_{t,I}$ of the implementation I is an estimation of the power consumption value $\mathcal{P}_{\mathcal{L}_{t,I}}$, which bases upon

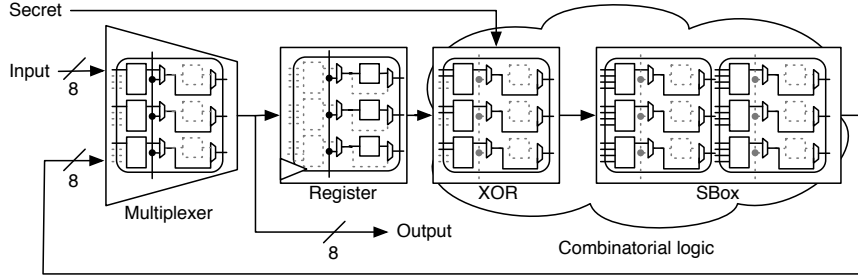


Figure 4.1: AES circuit example

hypothetical values of \mathbf{y} , symbolized by $\tilde{\mathbf{y}}$. For every assumed $\tilde{\mathbf{y}}$ value of the set Υ^m ³, $\tilde{\mathcal{L}}_{t,I}$ is determined by using the same fixed set of \mathbf{x} values for different values of $\tilde{\mathbf{y}}$. All estimated $\tilde{\mathcal{P}}_{\mathcal{L}_{t,I}}$ values are grouped by the used $\tilde{\mathbf{y}}$ values. This procedure is repeated for all i different \mathbf{y} values, which are used during the acquisition phase. In the evaluation process a side-channel distinguisher $\mathcal{D}(\mathcal{L}_{t,I}, \tilde{\mathcal{L}}_{t,I})$ compares the captured power consumption of the implemented circuit $\mathcal{P}_{\mathcal{L}_{t,I}}$ to the estimated hypothetical power consumption $\tilde{\mathcal{P}}_{\mathcal{L}_{t,I}}$. The group with the most likely same $\tilde{\mathcal{P}}_{\mathcal{L}_{t,I}}$ values as the captured $\mathcal{P}_{\mathcal{L}_{t,I}}$ is ranked at the top and thus refers to the most likely $\tilde{\mathbf{y}}$. Thus, a fully exploitable $\mathcal{L}_{t,I}$ of the design yields $\tilde{\mathbf{y}} = \mathbf{y}$.

4.1.2 Evaluation Metrics

Several different side-channel metrics have been proposed in literature in the last years with the goal to pinpoint the side-channel resistance of an implementation or to compare different side-channel distinguishers. The most frequently used metrics in the area of power analysis will be discussed here in detail by means of two, rather simple circuits. These circuits are “toy” versions of AES and functionally describe the processing of just one byte at the last round of AES as depicted in Fig. 4.1. Therefore, the secret round key k and the input p are of 8 bit width only and, thus, $\Upsilon^m = \Upsilon$, $\mathcal{X}^n = \mathcal{X}$ and $\mathbf{y} = k \in \{0, \dots, 255\}$, $\mathbf{x} = p \in \{0, \dots, 255\}$ hold in this case, cf. Def. 3. Both versions are identical, except for the implementation of the SBox. In one case a straight forward, algorithmic implementation of a composite Field SBox *COMP* is realized according to the description in [SMTM01]. The other SBox *PPRM3* is realized from an implementation published in [MS02] and is a three stage *XOR-AND* representation of a composite field SBox.

The small difference of the SBox structures in the designs affects the amount of exploitable leakage at a certain degree of effort. All metrics utilize the sorting

³represents the search space of all values of \mathbf{y}

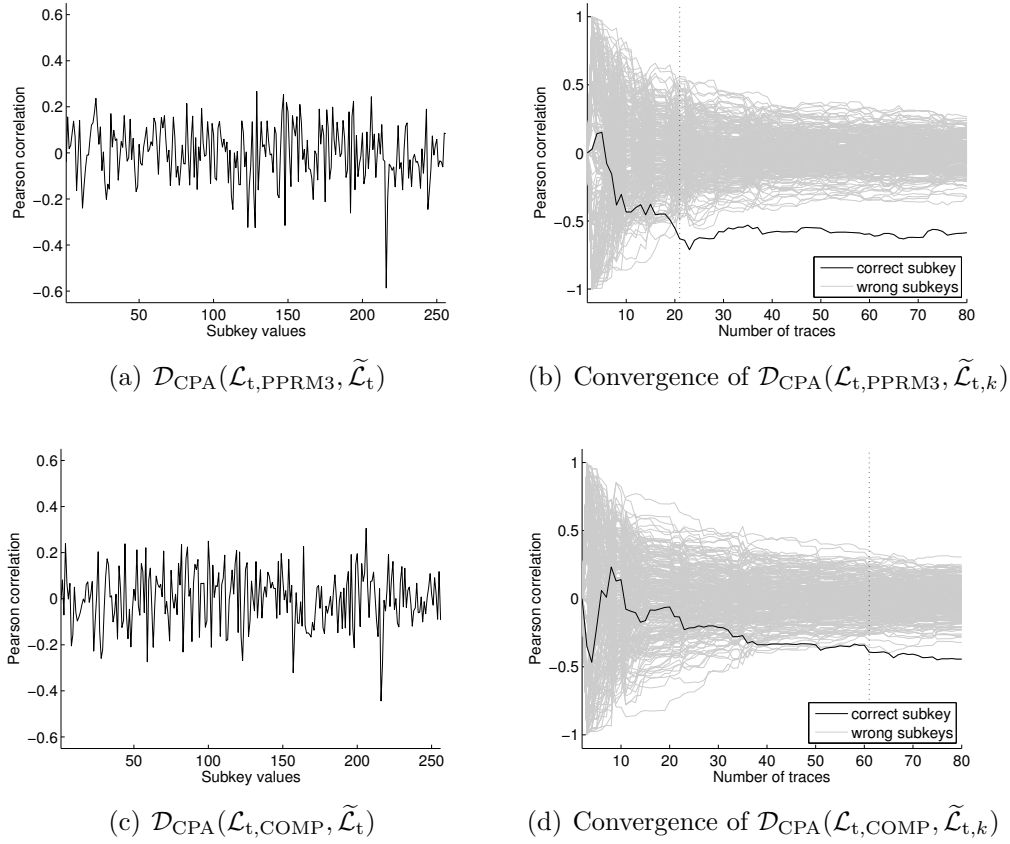


Figure 4.2: Correlation result: All key values and their convergence over the number of traces are depicted (a), (b) for the PPRM3 circuit and in (c), (d) for the COMP circuit, respectively.

results of the same side-channel distinguisher in order to get the vulnerability level of these circuits. For a better understanding of the origin of the identified leakage of the corresponding circuit, the information quality of the results of the evaluation metric is of special interest. In the illustrated example the correlation-based distinguisher was used, which was introduced by [BCO04] as part of the CPA method. For both designs the same $\mathcal{L}_{\mathcal{M}_t}$ was exploited. This model stems from a distance model, which exploits the value transitions inside the register of the circuit. Thus, the resulting leakage function is given by:

$$\mathcal{L}_{\mathcal{F}_{t,\text{toy-AES}}}(k, p) = \sum_{i=1}^8 (p \oplus \text{SBox}(p \oplus k))_i \quad (4.2)$$

The result produced from the inner term is summed up bitwise to calculate the Hamming distance of the value transition inside the register, cf. Hamming distance in Eq. 4.1. $\mathcal{L}_{\mathcal{F}_t}(k, p)$ is used to estimate $\tilde{\mathcal{L}}_t$ for the all possible k values.

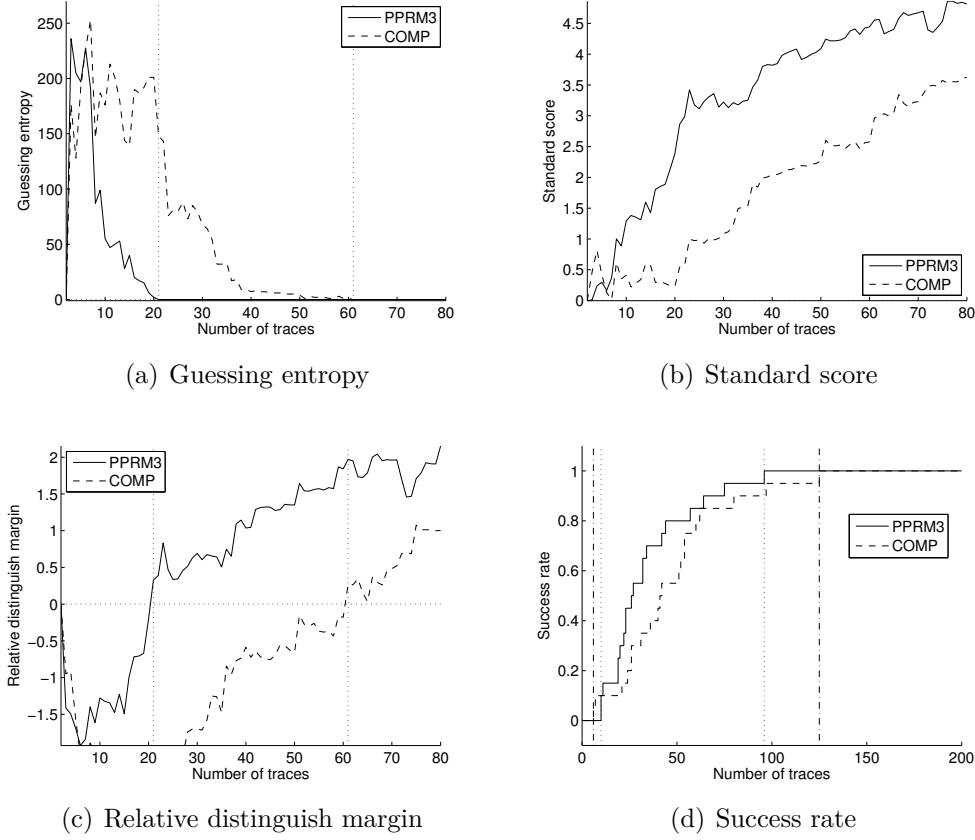


Figure 4.3: Results of the different evaluation metrics based on the outcome of the distinguisher, as depicted in Fig. 4.2.

Figure 4.2 depicts the results gained from $\mathcal{D}_{\text{CPA}}(\cdot, \cdot)$ with an used set for 100 captured traces of each implementation. These results are the basis for the discussion of the following side-channel metrics. Various metrics will be introduced and their quality of information for the design process will be discussed.

Guessing entropy The guessing entropy provides the information of how many key values have to be tested after the performed side-channel evaluation. This metric simply delivers the position of the correct $\tilde{\mathbf{y}}^c = \mathbf{y}^c$ out of a list of all $\tilde{\mathbf{y}}$ ranked by the distinguisher. Figure 4.3(a) clearly shows the different numbers of needed traces to get $\mathcal{D}_{\text{CPA}}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t, k_{(0)}^c})$ of both designs ranked at position 1. Guessing entropy was originally introduced in [KB07] as an information theoretical metric and was applied as a practical metric for side-channel evaluation by [SMY09]. However, this metric does not provide any useful information about how good the hypothetical leakage $\tilde{\mathcal{L}}_t$ based on $\tilde{\mathbf{y}}^c$ is distinguishable from its nearest neighbors.

Standard score The standard score as given in Eq. (4.3) was introduced in [GHP04] and is similar to the signal-to-noise ratio (SNR) mentioned in [MPO07]. Compared to the guessing entropy this metric provides the information on how distinguishable the exploitable \mathcal{L}_t is from noise. The down side is, as Fig. 4.3(b) depicts, that it is not directly readable at what position the correct $\tilde{\mathbf{y}}^c = \mathbf{y}^c$ is ranked or if it is at the first position at all. Thus, a high standard score does not need to reveal the correct \mathbf{y}^c and thus does not necessarily lead to a successful exploitation of \mathcal{L}_t .

$$\text{Standard Score} = \frac{\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}^c}) - \mathbb{E}(|\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}})|) \mid \forall \mathbf{y}}{\sqrt{\text{Var}(\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}})) \mid \forall \mathbf{y}}} \quad (4.3)$$

Relative distinguish margin This metric was introduced in [WO11b]. It combines the guessing entropy and the standard score to a certain degree. Equation 4.4 formalizes the distance based evaluation metric.

$$\text{Relative Distinguish Margin} = \frac{|\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}^c})| - \max(|\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}})| \mid \forall \mathbf{y} \neq \mathbf{y}^c)}{\sqrt{\text{Var}(\mathcal{D}_{\text{CPA}}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}})) \mid \forall \mathbf{y}}} \quad (4.4)$$

The distance between the highest ranked alternative $\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}})$ value and the value of the $\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}^c})$ is caused by the correctly estimated secret parameter \mathbf{y}^c . This distance is normalized by the standard deviation over all $\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}})$ including the correct $\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,\mathbf{y}^c})$ value. If the relative distinguish margin is above zero, then the \mathcal{L}_t is clearly exploitable and a side-channel attack would be successful. A failing side-channel attack would be indicated by a value below zero. Therefore, this metric provides the needed information for the design process to differ between the design's side-channel properties, cf. Fig. 4.3(c).

Success rate The success rate returns the average probability that the correct secret value \mathbf{y}^c is ranked on the first position on a set of side-channel attacks with an individual set of captured traces for each attack. As depicted in Fig. 4.3(d), the more side-channel resistant an implementation is, the more traces are required to reach the 100% boundary. The figure displays the success rate over 50 CPA attack experiments for both designs, i.e., the PPRM3 and the COMP. Thus, this metric is very similar to the guessing entropy method, but compared to that metric it considers the average vulnerability of an implementation. The designer deduces the same information from the success rate as from the guessing entropy.

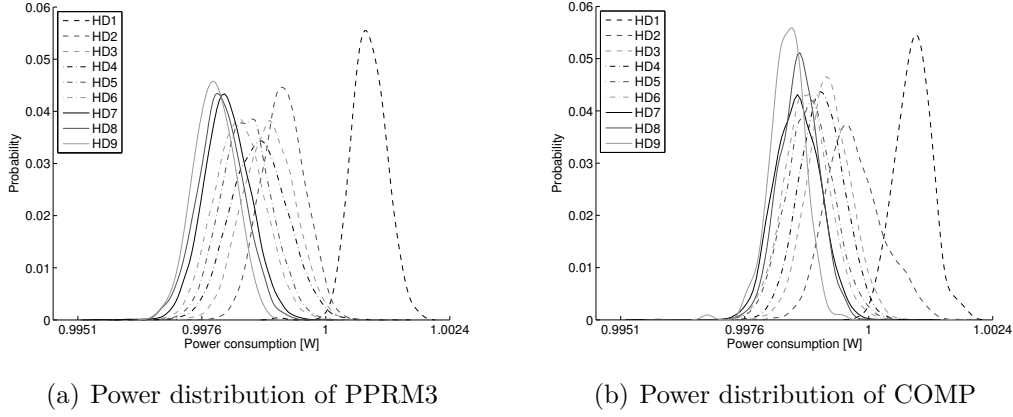


Figure 4.4: Template-based power consumption characterization of the implementations given in Fig. 4.1

From all the discussed side-channel evaluation metrics the relative distinguish margin seems most promising for the purpose of quantifying different implementations of the same functionality. Besides the information of how many traces are required to exploit the \mathcal{L}_t in order to compromise the circuits secret, the metric also provides the insight of the SNR. Thus, it yields directly how good the $\tilde{\mathcal{L}}_{t,y}$ of the correct secret value y^c is distinguishable from all the other $\tilde{\mathcal{L}}_{t,y}$. Another metric, which might be useful, is the mutual information. The mutual information is mainly used as a side-channel distinguisher, the so-called MIA, cf. [GBTP08]. In [SMY09] it is proposed as a means to compare the vulnerability of different designs according to a certain attack hypothesis. Nevertheless, these authors argue that less entropy not always yields a better success rate. In this case MIA is not directly used as a metric, because profiling-based approaches are used to determine the vulnerability of the implementation. However, a non-linear model seems to be more efficient than the combination of a linear model and a non-linear distinguisher, cf. [WOS12].

4.2 CSA-Based Evaluation Metric

The discussed metrics in Subsect. 4.1.2 utilize the results of the selected side-channel distinguisher in order to provide the essential information from the distinguisher results. This information yields the distinguishability between all $\tilde{\mathcal{L}}_{t,\tilde{y}} \mid \forall \tilde{y} \neq y^c$ and $\tilde{\mathcal{L}}_{t,y^c}$, which best fits the emitted \mathcal{L}_t of the implementation. The confidence of those outcomes are premised on the result of $\mathcal{D}(\mathcal{L}_t, \tilde{\mathcal{L}}_{t,y^c})$, which strongly depends on the selected $\mathcal{L}_{\mathcal{M}_t}$ and $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$. According to that circumstances it is crucial to select both an appropriate $\mathcal{L}_{\mathcal{M}_t}$ and an appropriate

$\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$. None of the previously discussed metrics provide the insight about the quality of the selected $\mathcal{L}_{\mathcal{M}_t}$ or even the origin of the side-channel leakage in terms of the selected $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$. A straight-forward approach to cope with the problem of selecting appropriate $\mathcal{L}_{\mathcal{M}_t}$ and $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ in order to generate adequate $\tilde{\mathcal{L}}_{t, \tilde{\mathbf{y}}}$ is to use the template method firstly proposed in [CRR02]. As depicted in Fig. 4.4, this method constructs a distribution of the power consumption for every parameter set or, like presented in here, based on a simple $\mathcal{L}_{\mathcal{M}_t}$ such as HD.

From these distributions it is clearly visible that some groups of Hamming distances are more easy to distinguish than others, which yields a better exploitation of the \mathcal{L}_t . But still there is neither a constructive feedback on the origin of the side-channel leakage of these implementations nor a quantification of the assumed \mathcal{L}_t . A concept, which satisfies the request for a quantitative analysis aimed to an identification of the side-channel origin inside a circuit, was introduced recently. The concepts of the so-called *constructive side-channel analysis* for FPGA platforms was firstly mentioned in a previous joint publication [KSS10]. In this cooperative work the author's own contribution was to empirically verify the theoretical concepts of the quantitative feedback between the $\mathcal{L}_{\mathcal{M}_t}$ and the design. This includes the adaption of the theoretical approaches to a practical example in terms of interpreting the results of the constructive side-channel analysis and to compare them to an in-depth analysis of the example circuit. During this empirical verification it was demonstrated by comparison to another design that a different design with the same functionality yields completely different results from the constructive side-channel analysis. Thus, this thesis exploits these basic concepts and extends them to cover the focused purpose.

4.2.1 Constructive Side-Channel Analysis

The core idea of the constructive side-channel analysis (CSA) is to gain insight into implementation flaws after or during the construction phase of a design. One of the central elements in the proposed approach is to provide quantitative information on the side-channel leakage of a design, which can be used in a constructive manner to improve the circuit subsequently. The quality of the evaluation results based on a CSA is improved by providing additional information about the quantification of the exploitable vulnerabilities of the analyzed design. Due to an extended profiling-based $\mathcal{L}_{\mathcal{M}_t}$ the model properties unveil the origin of the side-channel leakage. Additionally, it was demonstrated in the preliminary joint publications [HKSS11] and [HSS12] that CSA offers verification methods in order to check the $\mathcal{L}_{\mathcal{M}_t}$ suitability and accuracy. Thus, the usage of a CSA-based approach truly supports the evaluation of runtime mutating countermeasures against power analysis attacks in terms of quantifying the side-channel

resistance improvement provided by these methods.

The first phase of the Stochastic Approach⁴ (SA) is used as the basis for CSA. Originally, this approach was developed to attack block cipher schemes by exploiting the data-dependent electrical current consumption. This profiling-based method was developed to exploit side-channel information more efficiently than non-profiling based attack methods. The improvement can be equally efficient to classical template attack, but with less profiling effort, cf. [SLP05, LRP07, Sch08]. The efficiency of the Stochastic Approach is comparable to the efficiency of template attacks especially for a small set of profiling traces, cf. [GLRP06].

For designing side-channel resistant circuits one of the strongest possible attack has to be considered in the evaluation phase in order to get the correct confidence level of security of the implementation and its vulnerabilities. Thus, it is crucial to utilize profiling-based evaluation methods in order to gain a better knowledge of the vulnerability than in case of a non-profiling analysis. The SA is more suitable as the underlying means for the CSA methods than the classical template attacks, because it gains more quantitative insight. Due to the adaptation of the $\mathcal{L}_{\mathcal{M}_t}$ in the profiling phase via a linear regression method, the impact of the different leakage assumptions formulated as $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ is directly yielding the origin of the \mathcal{L}_t inside the circuit. Therefore, the result of the CSA is more transparent with respect to the circuit activity in combination with the exploitable side-channel leakage than the classical template method introduced by [CRR02]. This transparent quantitative relationship between the cause of the \mathcal{L}_t inside the circuit and the \mathcal{L}_t itself is of great interest for investigating especially the impact of mutating countermeasures on the side-channel resistance. In the following a short outline of the CSA approach will be given in conjunction to the Stochastic Approach.

CSA based on the first phase of the Stochastic Approach Typically, like all other side-channel attack methods, the SA utilizes the *divide-and-conquer* principle. The realization of the random variables for the subkeys are denoted by k and the plaintext or ciphertext is denoted by x . In order to indicate the bit width s and p of these values the following properties are applied: $k \in \{0, 1\}^s$ and $x \in \{0, 1\}^p$, cf. [Sch08]. In this thesis, the meaning of the designator \mathbf{x} still holds, while k equals to the denotation \mathbf{y} regarding Def. 3. The captured physical observable, i.e., the power consumption⁵, is interpreted as a random variable, which is sampled precisely at the moment when the traces are captured by a measurement device. These sampled physical instants $\mathcal{P}_t(\mathbf{x}, \mathbf{y})$ at a certain point in time t are a composition of a deterministic part $h_t(\mathbf{x}, \mathbf{y})$ and normally

⁴see Subsect. 2.2.1.2

⁵originally the electrical current consumption

distributed noise \mathcal{R}_t ⁶. Thus, the general assumption of the $\mathcal{L}_{\mathcal{M}_t}$ of the SA is:

$$\mathcal{P}_t(\mathbf{x}, \mathbf{y}) = h_t(\mathbf{x}, \mathbf{y}) + \mathcal{R}_t \quad (4.5)$$

The distribution of $\mathcal{P}_t(\mathbf{x}, \mathbf{y})$ depends on the tuple (\mathbf{x}, \mathbf{y}) , but it is unknown. The parameter (\mathbf{x}, \mathbf{y}) affects the also unknown deterministic additive component of the power consumption $h_t(\mathbf{x}, \mathbf{y})$. This term $h_t(\mathbf{x}, \mathbf{y})$, in [KSS10] denoted as *leakage function*, refers directly to the exploitable \mathcal{L}_t of the analyzing circuit, because this component is the only deterministic element of the unknown distribution $\mathcal{P}_t(\mathbf{x}, \mathbf{y})$ that responses to the algorithm parameters (\mathbf{x}, \mathbf{y}) . The unknown function $h_t(\mathbf{x}, \mathbf{y})$ equals to $\mathcal{L}_{\mathcal{F}_{t, \text{CSA}}}(\mathbf{x}, \mathbf{y})$ and quantifies the $\mathcal{L}_{\mathcal{M}_{t, \text{CSA}}}$. The task of the linear regression in the profiling phase is then to estimate this unknown function $h_t(\mathbf{x}, \mathbf{y})$ by an approximator, denoted as $\tilde{h}_t(\mathbf{x}, \mathbf{y})$. The core idea is to represent $\tilde{h}_t(\mathbf{x}, \mathbf{y})$ as a function for mapping every value of \mathbf{y} in a 2^p -dimensional subspace $\mathcal{F}_{\mathbf{y}}$ (where p is the bit-width of \mathbf{x}). These functions directly map all tuple combinations of \mathbf{x} and \mathbf{y} onto \mathbb{R} :

$$\mathcal{F}_{\mathbf{y}} := \{h' : \{0, 1\}^p \times \{\mathbf{y}\} \rightarrow \mathbb{R}\} \quad (4.6)$$

The ‘clue’ of this approach is that the search space for the estimated leakage function $\tilde{h}_t(\mathbf{x}, \mathbf{y})$, represented as a subspace $\mathcal{F}_{\mathbf{y}}$, can be reduced to the relevant and, hence, the appropriate dimensions. Thus, the approach is aiming at the best approximator $\tilde{h}_t^*(\mathbf{x}, \mathbf{y})$ for the unknown leakage function $\mathcal{L}_{\mathcal{F}_{t, \text{CSA}}}(\mathbf{x}, \mathbf{y})$ described by an u -dimensional subspace $\mathcal{F}_{u, t; \mathbf{y}}$. This subspace is spanned up by u basis functions $g_{j, t; \mathbf{y}}$ with $j = 0, \dots, u - 1$:

$$\begin{aligned} \mathcal{F}_{u, t; \mathbf{y}} := \quad & \{h' : \{0, 1\}^p \times \{\mathbf{y}\} \rightarrow \mathbb{R} \mid \\ & h' = \sum_{j=0}^{u-1} \beta'_j g_{j, t; \mathbf{y}} \text{ with } \beta'_j \in \mathbb{R}\}. \end{aligned} \quad (4.7)$$

The functions $g_{j, t; \mathbf{y}}$ represent the basis vectors of the subspace $\mathcal{F}_{u, t; \mathbf{y}}$ and should cover the assumed relevant sources of the implementation \mathcal{L}_t , cf. [KSS10]. The impact of the basis functions to the correct estimation of $\tilde{h}_t(\mathbf{x}, \mathbf{y})$ is scaled by the value of the weighting coefficients β' in Eq. 4.7. Thus, $\mathcal{L}_{\mathcal{F}_{t, \text{CSA}}}(\mathbf{x}, \mathbf{y})$ can be estimated depending on each basis function by determining its β -coefficient by means of a linear regression. This regression utilizes the profiling phase of the SA, in which a set of N_1 measurements is captured with known values for \mathbf{y} and \mathbf{x} at a certain point in time. It is proceeded in that manner so that the measured power consumptions $\mathcal{P}_t(\mathbf{x}, \mathbf{y})$ can be arranged within a vector of the

⁶it is assumed that the distribution is normally distributed with $E(\mathcal{R}_t) = 0$ and is independent from $h_t(\mathbf{x}, \mathbf{y})$

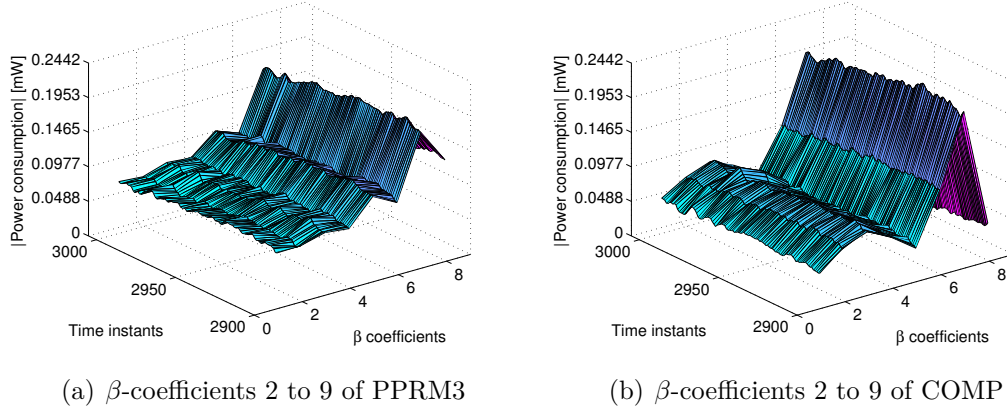


Figure 4.5: Results of the CSA evaluation on the AES example circuits by utilizing a $\mathcal{F}_{9,t;\mathbf{y}}$ as base for the $\mathcal{L}_{\mathcal{M}_t}$

dimension $(N_1 \times 1)$. Based on the used \mathbf{x} values for the N_1 measurements, a $(N_1 \times u)$ -matrix is then composed:

$$\mathbf{A} := \begin{pmatrix} g_{0,t;\mathbf{y}}(\mathbf{x}_1, \mathbf{y}) & \cdots & g_{u-1,t;\mathbf{y}}(\mathbf{x}_1, \mathbf{y}) \\ \vdots & \ddots & \vdots \\ g_{0,t;\mathbf{y}}(\mathbf{x}_{N_1}, \mathbf{y}) & \cdots & g_{u-1,t;\mathbf{y}}(\mathbf{x}_{N_1}, \mathbf{y}) \end{pmatrix}. \quad (4.8)$$

Usually, \mathbf{A} contains the switching activity of each of the basis vectors of the subspace. By utilizing the linear regression the equation system $\mathbf{A} \mathbf{b} = \mathcal{P}_t(\mathbf{x}, \mathbf{y})$ has the solution

$$\tilde{h}_{t;\mathbf{y}}^*(\cdot, \mathbf{y}) = \sum_{j=0}^{u-1} \tilde{\beta}_{j,t;\mathbf{y}}^* g_{j,t;\mathbf{y}}(\cdot, \mathbf{y}) \quad \text{with } \tilde{\beta}_{j,t;\mathbf{y}}^* := \tilde{b}_j^*, \quad (4.9)$$

which is estimated by the least square method of $h_{t;\mathbf{y}}^*$. In the preliminary work [KSS10] it was demonstrated that the β -characteristic provides quantitative feedback and is applicable for (re-)design methods. The own contribution to this publication was the major part of applying this theoretical concept to a practical FPGA implementation. So, the β -characteristic was exploited to methodically reveal the flaws of implementation.

$\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ for the AES example After this informative overview of the first phase of the Stochastic Approach enlightened in an constructive context (CSA), it is now demonstrated how to utilize this approach for the example circuits. Afterwards, improvements of the CSA concept with new methods and models are introduced. Since then, the weighting coefficients of the linear regression are referred to as β -coefficients and are not further specified in a physical context.

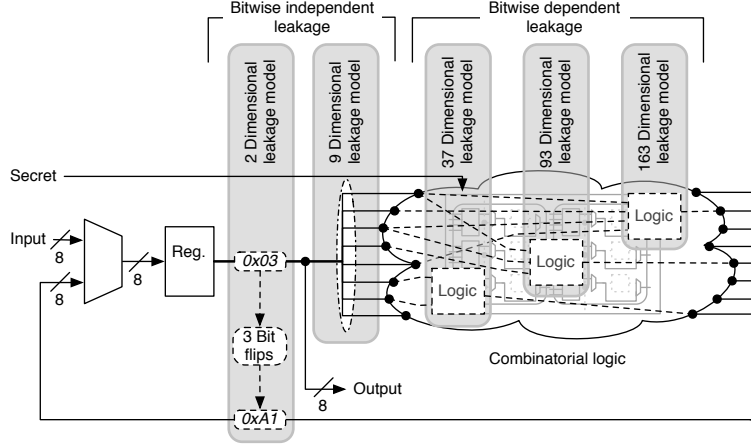


Figure 4.6: Scheme of the effects captured by different leakage models

At a closer look these β -coefficients are the averaged amount of consumption per basis of the subspace of the measured medium, for instance the power consumption. For the 2-dimensional $\mathcal{F}_{2,t;y}$ the basis 2 directly refers to the consumption of a HD or HW approach. Concerning $\mathcal{F}_{9,t;y}$, the basis 2 to 9 refer to the bit lines independently and thus the averaged sole consumption of each bit line is depicted.

In case of analyzing toy-AES⁷ circuits from Subsect. 4.1.2 the Hamming distance based $\mathcal{L}_{\mathcal{F}_{t,\text{toy-AES}}}(k, p)$ from Eq. (4.2) refers to the following $g_{j,t;y}(\mathbf{x}, \mathbf{y})$ of the 2-dimensional $\mathcal{L}_{\mathcal{F}_{t,\text{toy-AES,CSA}^2}}(k, p)$:

$$\begin{aligned} g_{0,t;k}(p, k) &= 1 \\ g_{1,t;k}(p, k) &= \left(\sum_{i=1}^8 (p \oplus \text{SBox}(p \oplus k))_i \right) - 4 \end{aligned} \quad (4.10)$$

As usual, the first basis of the subspace is set to the constant value of 1 to capture all non-zero expectation values of the measured power consumption by the least square estimation in order to fulfill the requirements $E(\mathcal{R}_t) = 0$ and $E(g_{j,t;y}(\cdot))|_{\forall j \neq 0} \neq 0$. The subtraction of four for the second basis vector assures that the basis is at least normalized, cf. [Sch08]. The depicted results of Fig. 4.5 stem from the 9-dimensional $\mathcal{F}_{9,t;y}$, which is analogously constructed to the subspace $\mathcal{F}_{2,t;y}$. The $\mathcal{L}_{\mathcal{F}_{t,\text{toy-AES,CSA}^9}}(k, p)$ of the subspace $\mathcal{F}_{9,t;k}$ refers to the following basis vector functions:

$$\begin{aligned} g_{0,t;k}(p, k) &= 1 \\ g_{j,t;k}(p, k) &= (p \oplus \text{SBox}(p \oplus k))_j - \frac{1}{2^1} \quad \text{for } j = 1, \dots, 8 \end{aligned} \quad (4.11)$$

⁷Note, the case specific setting of \mathbf{x} and \mathbf{y} in the beginning of SubSect. 4.1.2

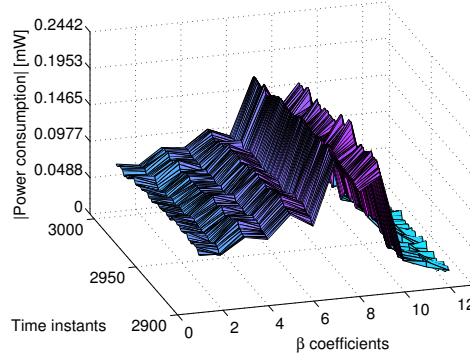


Figure 4.7: β -coefficients of $\mathcal{L}_{\mathcal{F}_{t, \text{toy-AES}, \text{CSA}^{12}}}(k, p)$ exploiting basic FPGA components

As before, it is ensured that basis vectors have an expectation value of zero by the subtraction of $\frac{1}{2^1}$ to normalize the basis. The shapes of the plots of the absolute value of the β -characteristic in Fig. 4.5(a) and Fig. 4.5(b) clearly indicate the different implementation properties of these two functionally identical designs. These figures provide the information that for both designs the seventh bit has the biggest impact on the power consumption. While the COMP circuit has only two significant bits yielding a significant power consumption, the PPRM3 design has more significant bits with a higher consumption. This information is not provided by the result of the distribution analysis depicted in Fig. 4.4. However, the absolute values of the plots cannot be compared directly, because the β -coefficient values depend on the absolute value of the overall captured power consumption value $\mathcal{P}_t(\mathbf{x}, \mathbf{y})$. Thus, the β -characteristic of a design is not directly comparable with any other design in terms of side-channel resistance.

In the preliminary joint work of [KSS10] the relationship between this significant position in the plot and the more power consuming structure of the circuits was validated by the thesis author. In this, as in all the other joint publications relating the SA, the value transitions inside registers were always exploited in order to construct the activity matrix \mathbf{A} , cf. Eq. (4.8). Thus, the power consumption of the switching activity and the different load properties of the bit lines to the registers were exploited bitwise only.

As suggested in [DPRS11], the leakage may also arise from interactions within the circuit between two or even more bit lines. The so-called *higher dimensional subspace based leakage models* are more reasonable, if some additional switching activity of the combinatorial circuit, for instance hazards, occurs, which depends on several parameters, cf. Fig 4.6. In the joint preliminary work [HKSS11] this property was investigated and the results clearly show that the \mathcal{L}_t based on the interaction between bit lines has a great impact on the side-channel resistance

of a design. More precisely, the subspace of dimension 37, 93, and 163 were investigated. The interaction of these subspaces is shown in the Fig. 4.6 and their leakage functions are listed in the appendix B.

Concerning the architecture of an FPGA-based design and the gained insight from higher dimensional subspace based leakage models following questions rise:

What degree of granularity does the CSA approach offer? Is it possible to model a $\mathcal{L}_{\mathcal{F}_t}(\mathbf{x}, \mathbf{y})$ to characterize the \mathcal{L}_t caused by a basic element of the FPGA, for instance a LUT?

To answer these questions in this thesis the $\mathcal{L}_{\mathcal{F}_{t, \text{toy-AES}, \text{CSA}^9}}(k, p)$ of the $\mathcal{F}_{9, t; \mathbf{y}}$ is extended by 4 additional subspaces, which each holds the boolean description of a basic element of the FPGA. For this experiment lookup tables of the SBox circuit were selected, which are directly connected to the input of the SBox area of the circuit. The originally exploited value transitions inside the circuit described by $\mathcal{L}_{\mathcal{F}_{t, \text{toy-AES}, \text{CSA}^9}}(k, p)$, as denoted in Eq. 4.2, are adapted to the transitions of the value changes at the SBox input. Thus, the basis vectors of $\mathcal{L}_{\mathcal{F}_{t, \text{toy-AES}, \text{CSA}^{12}}}(k, p)$ are defined as:

$$\begin{aligned} g_{0, t; k}(p, k) &= 1 \\ g_{j, t; k}(p, k) &= ((p \oplus k) \oplus \text{SBox}(p \oplus k))_j - \frac{1}{2^1} \quad \text{for } j = 1, \dots, 8 \\ g_{j, t; k}(p, k) &= F_{\text{lut}_{j-8}}((p \oplus k) \oplus \text{SBox}(p \oplus k))_j \quad \text{for } j = 9, \dots, 12. \end{aligned} \tag{4.12}$$

The coefficients to weight the switching activity describing functions are moved to the lookup table functions (F_{lut}), which are presented in more detail in the Appendix B. The results in Fig. 4.7 depict that only one of the basis vectors, which models the LUT switching activity, has an equally significant impact to the solely bit transition oriented basis vectors 2 to 9. Nevertheless, the empirical validation of the possibility to characterize or to measure the vulnerability of basic FPGA components by the CSA approach is a very useful feature of the CSA. This feature indeed provides the possibility to specify the required structural changes by means of a mutation of the data path, while the functional behavior is still maintained by the modified circuit.

4.2.2 Signal-To-Noise Based Evaluation Metric

A common method to characterize the quality of a measured trace is the signal-to-noise ratio (SNR). It has a great influence on the success rate of a side-channel evaluation and thus is an important characteristic for security implementations, cf. [GHP04, MPO07, GMS⁺11]. For instance, the SNR may be used to quantify the strength of countermeasures. In this section it is introduced how to use the first phase of the SA to estimate the SNR. In particular, this method has the advantage that it is not constrained to some fixed leakage model (like the Hamming

Distance), it may be viewed as a means to combine the SNR with any appropriate leakage models. This metric also solves the incompatibility when comparing the weighting coefficients (β -coefficients) of two different designs. Thus, the SNR metric is an useful extension of the supportive CSA methods. As Fig. 4.8 shows, the two different implementations of the example circuit in Subsect. 4.1.2 are now directly comparable in terms of the relationship between the exploitable leakage \mathcal{L}_t and the noise \mathcal{R}_t at certain points in time.

SNR Leakage Estimation The signal-to-noise ratio is an important metric to distinguish a determinable signal or elements of a distribution of high interest from signals or elements of a distribution of no interest, which is often denoted as noise (\mathcal{R}). In general, the SNR is defined as

$$SNR = \frac{\text{Var}(\text{signal})}{\text{Var}(\text{noise})}, \quad (4.13)$$

which characterizes the ratio between the variance of the determinable signal or elements of a distribution and the noise of the measurement. A more accurate ratio for a power analysis orientated SNR is given in [MPO07] by

$$SNR = \frac{\text{Var}(\mathcal{P}_{\mathcal{L}_t})}{\text{Var}(\mathcal{P}_{\mathcal{O}\mathcal{P}_{t,\text{others}}} + \mathcal{P}_{\mathcal{R}_t})}, \text{ whereas } \mathcal{P}_t = \mathcal{P}_{\mathcal{L}_t} + \mathcal{P}_{\mathcal{O}\mathcal{P}_{t,\text{others}}} + \mathcal{P}_{\mathcal{R}_t}. \quad (4.14)$$

Comparing the SNR oriented standard score metric⁸ with the metric in Eq. (4.14), the SNR Eq. (4.14) is not constrained to a deterministic fixed-value of a side-channel distinguisher result. Thus, the metric in Eq. (4.14) is more suitable to be adapted to the linear regression result of the CSA.

Please note that the power consumption \mathcal{P}_t and its composing components are interpreted as a realizations of random variables, which are yielded from sampling these random variables at the measurement process. $\mathcal{P}_{\mathcal{L}_t}$ denotes the exploitable power consumption, which is approximated by the data-dependent deterministic part $h_{t;k}^*$. Furthermore, $\mathcal{P}_{\mathcal{R}}$ denotes the power consumption due to the noise, which is captured by \mathcal{R}_t . Finally, $\mathcal{P}_{\mathcal{O}\mathcal{P}_{t,\text{others}}}$ denotes the power consumption of parallel running activities of the circuit and, of course, the approximation error of the linear regression.

As mentioned before, SA originally exploits the electrical current consumption, which is proportional to the power consumption due to the measurement over an ohmic resistor shunt, c.f., Eq. (4.15). Of course, this also holds for the additive components of the measured power consumption. Similar to the description in the joint publication [HSS12], the following relation holds:

$$\frac{\text{Var}(I_{\mathcal{L}_t})}{\text{Var}(I_{\mathcal{O}\mathcal{P}_{t,\text{others}}} + I_{\mathcal{R}_t})} \sim \frac{\text{Var}(\mathcal{P}_{\mathcal{L}_t})}{\text{Var}(\mathcal{P}_{\mathcal{O}\mathcal{P}_{t,\text{others}}} + \mathcal{P}_{\mathcal{R}_t})}. \quad (4.15)$$

⁸mentioned in Subsect. 4.1.2

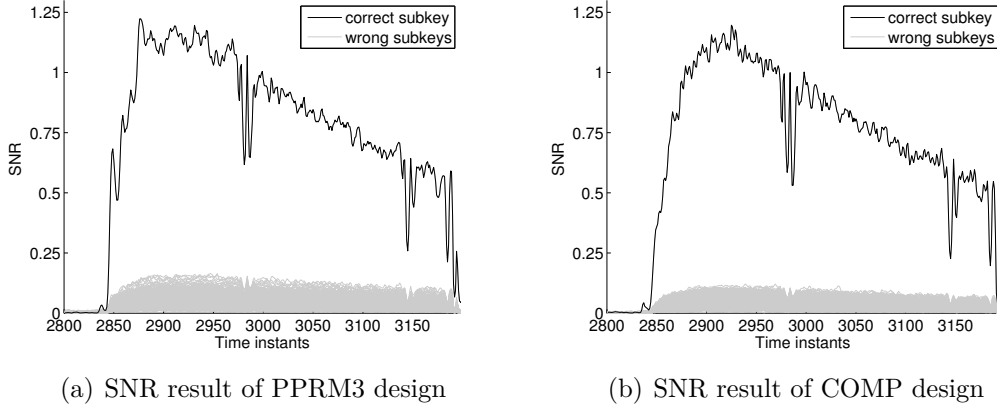


Figure 4.8: SNR-based CSA evaluation of different designs of the AES example

In this preliminary work the concept and the idea of utilizing a signal-to-noise ratio like metric to rate the $\mathcal{L}_{\mathcal{M}_t}$ accuracy for the CSA approach as well as a major part of the empirical validation was contributed by the thesis author. For the following equations it is considered that the mapping expressions of the functions $g_{0,t;\mathbf{y}}(\cdot, \mathbf{y}), \dots, g_{u-1,t;\mathbf{y}}(\cdot, \mathbf{y})$ construct a \mathcal{F}_u with an orthonormal basis in order to assure correct relationships between the different weighting coefficients. Straight-forward computations for the determinable information yield

$$\begin{aligned}
 \text{Var}(\mathcal{P}_{\mathcal{L}_t}) &= \text{Var}_X(h_{t;\mathbf{y}}^*(X, \mathbf{y})) \\
 &= E_X(h_{t;\mathbf{y}}^*(X, \mathbf{y})^2) - E_X^2(h_{t;\mathbf{y}}^*(X, \mathbf{y})) \\
 &= \sum_{j=0}^{u-1} \beta_{j,t;\mathbf{y}}^2 - \beta_{0,t;\mathbf{y}}^2 = \sum_{j=1}^{u-1} \beta_{j,t;\mathbf{y}}^2.
 \end{aligned} \tag{4.16}$$

The computation of the variance of the noise is done in a similar way,

$$\begin{aligned}
 \text{Var}(\mathcal{P}_{\mathcal{O}_{\mathcal{P}_t, \text{others}}} + \mathcal{P}_{\mathcal{R}_t}) &= \text{Var}_{X,R}(\Delta h_{t;\mathbf{y}}(X, \mathbf{y}) + R_t) \\
 &= \text{Var}_{X,R}(\mathcal{P}_t(X, \mathbf{y}) - h_{t;\mathbf{y}}^*(X, \mathbf{y})).
 \end{aligned} \tag{4.17}$$

The term $\Delta h_{t;k}(X, \mathbf{y})$ denotes the approximation error due to the least square estimation. Equation (4.18) follows from the fact $E_X(\mathcal{P}_{\mathcal{L}_t}(X, \mathbf{y}) - h_{t;\mathbf{y}}^*(X, \mathbf{y})) = 0$. Using (4.16) and (4.17) together, they result in an estimator for the signal-to-noise ratio

$$(\widetilde{SNR}) = \frac{\sum_{j=1}^{u-1} \widetilde{\beta}_{j,t;\mathbf{y}}^2}{\text{Var}(p_t(\mathbf{x}_1, \mathbf{y}) - \widetilde{h}_{t;\mathbf{y}}^*(\mathbf{x}_1, \mathbf{y}), \dots, p_t(\mathbf{x}_N, \mathbf{y}) - \widetilde{h}_{t;\mathbf{y}}^*(\mathbf{x}_N, \mathbf{y}))}. \tag{4.18}$$

$\text{Var}(\cdot)$ denotes the variance, which is taken over the distance between the estimated exploitable power consumption by $h_{t,k}^*$ and the realization of the power

Table 4.1: Comparison between success rate ratio and \widetilde{SNR} ratio

Design type	PPRM3	COMP	Ratio
# Traces for success rate=1 based on $\mathcal{D}_{\text{CPA}}(\mathcal{L}, \tilde{\mathcal{L}})$ at time instant 2877	96	125	0.768
SNR level at time instant 2877	1.223	0.932	$1,309 = \frac{1}{0,763}$

consumption p_t (due to sampling in the measurement process) over a set of N elements. For side-channel evaluation the \widetilde{SNR} quantifies the relation between exploitable information and the 'sum' of the power consumption $\mathcal{P}_{\mathcal{O}\mathcal{P}_{t,\text{others}}}$ of other circuit activities that run in parallel and the noise \mathcal{R}_t . Equation (4.18) requests an orthonormal base such as for the $\mathcal{F}_{9,t,k}$. If the selected subspace of the $\mathcal{L}_{\mathcal{M}_t}$ is not directly orthonormalizable, the following equation can be used:

$$(\widetilde{SNR}) = \frac{\text{Var}_X(h_{t,y}^*(X, \mathbf{y}))}{\text{Var}(p_t(\mathbf{x}_1, \mathbf{y}) - \tilde{h}_{t,y}^*(\mathbf{x}_1, \mathbf{y}), \dots, p_t(\mathbf{x}_N, \mathbf{y}) - \tilde{h}_{t,y}^*(\mathbf{x}_N, \mathbf{y}))}. \quad (4.19)$$

In case of comparing several subspaces, this method quantifies how much useful side-channel information the corresponding leakage model extracts from the traces in relation to the existing noise.

In [HSS12] it was demonstrated that the \widetilde{SNR} relationship between two different evaluation runs yields the same relation between the success rates. In that case the same captured traces were evaluated with different leakage models $\mathcal{L}_{\mathcal{M}_t}$. Applying the \widetilde{SNR} metric in the same context to the two different example test circuits from Sect. 4.1.2, again the metric approximated the coefficient of the success rate to each other quite well, cf. Tab. 4.1. In case of approximation the affect of the SNR on the success of the attack compared with an attack under idle SNR condition, the derivation the SNR relationship based on CPA, provided in [MPO07], could be affirmed experimentally. The success rate values of this experiment stems from the results depicted in Fig. 4.3(d), while the values of the \widetilde{SNR} stem from the results displayed in Fig. 4.8. The fourth column of Tab. 4.1 displays the relationship between the two metrics. They are inverse proportional to each other as observed empirically:

$$\frac{\text{Success rate}(\mathcal{D}_{\text{CPA}}(\mathcal{L}_{t,\text{PPRM3}}, \tilde{\mathcal{L}}_{t,\text{PPRM3}}))}{\text{Success rate}(\mathcal{D}_{\text{CPA}}(\mathcal{L}_{t,\text{COMP}}, \tilde{\mathcal{L}}_{t,\text{COMP}}))} \sim \frac{\widetilde{SNR}(\text{COMP})}{\widetilde{SNR}(\text{PPRM3})} \quad (4.20)$$

The small difference in the coefficient values seems to result from approximation errors of the linear regression within CSA.

The SNR metric provides the information on how good the exploitable information extracted from the captured power traces based on the learned linear regression model is, which is established at the end of the first phase of the stochastic approach. This is an advantage for using profiling based attacks with a powerful learning phase, because a designer can directly see the effect of his countermeasure directly by the SNR value. In addition, due to the linear regression coefficient the designer gets an quantitative feedback about the contribution of the exploitable power consumption per basis vector function, which can be used to model a specific property of the circuit. Especially for hiding based methods this metric can be utilized for the design evaluation of countermeasures in order to determine how much the distinguishability between exploitable leakage information and noise is decreased. Another advantage is that, due to the variance relation analysis of the current consumption based on the linear regression model and its subtraction of the actual captured current consumption of the targeted device, different implementations can be compared more fairly.

4.3 Acquisition Setup

The required physical determination of \mathcal{L}_t in the second side-channel evaluation phase⁹ is provided by an acquisition setup. The acquisition setup as detailed in this thesis is an electric laboratory measurement setup and consists of a platform target device, a digital oscilloscope, a laboratory power supply, and a personal computer, cf. Fig. 4.9. The power supply is used to provide a direct current source of appropriate quality. Thus, the power traces are actually measured from the target platform running the design under test implementation in normal operation mode. The need for physically capturing the exploitable power consumption is caused by the circumstances that the vendor of the target platform (Xilinx, Inc.) does not provide the needed accurate models. The supported power consumption models by Xilinx and the architectural insight at the simulation tools level are not precise enough for the purpose of side-channel evaluation of different implementations at the accuracy degree, which this thesis is investigating.

In order to reduce the systems methodical measurement errors in terms of measurement noise, the complete setup was composed with low noise components. Thus, the measurement platform the SASEBO-GII evaluation FPGA board [SAS] was used in this thesis, which has been meanwhile established as the standard platform for power analysis in the community of side-channel analysis. The utilized oscilloscope in the measurement setup provides a high sample

⁹see Subsect. 4.1.1

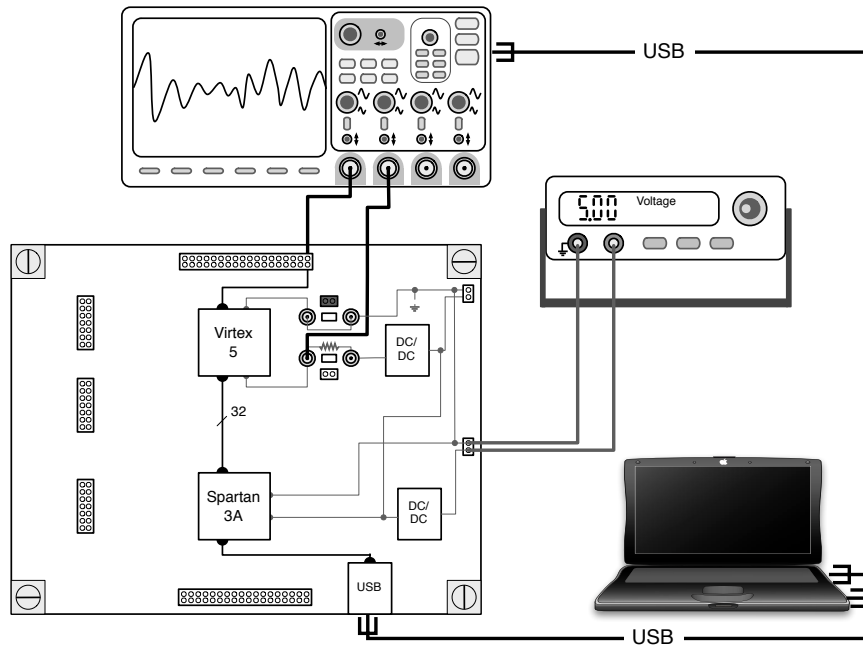


Figure 4.9: Acquisition setup

frequency of 500 MHz and a sample rate of 2 Gsa/s¹⁰ to quantify the power consumption at a sufficient resolution. The evaluation platform is separated into segments. One segment establishes the communication with the host, while the other one is considered for the side-channel measurement exclusively. These segments are electrically separated and only connected by a filtered bus connection between the FPGA of each segment. Beside a separate voltage supply and stabilizer the SASEBO-GII board offers measurement points for accessing the power consumption values. The power consumption is captured via a shunt measurement over the Virtex 5 target FPGA, cf. Fig. 4.9. A 1 Ω shunt resistor is exploited to measure the voltage basis to calculate the power consumption via the ohmic law. Please note that the power consumption is directly proportional to the measured voltage drop due to shunt measurement over an ohmic resistor.

Test environment on the Sasebo-GII The environment to test implementations on the evaluation board has been modified from the original version of the board manufacturer and a new architecture has been developed. Based on the results of a supervised bachelor thesis [Tha09] a transparent modular master-slave based communication architecture has been established. One request on the architecture was to provide an efficient and reliable communication

¹⁰Giga samples per second

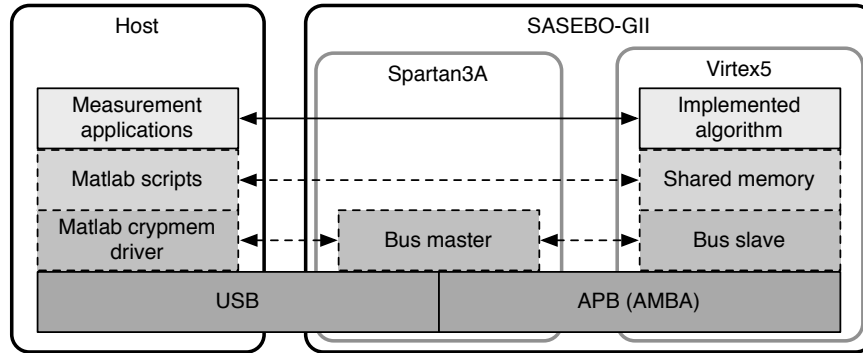


Figure 4.10: Layer model of the advocated communication between the host and the implemented algorithm

protocol. Thus, the interconnection of the two FPGAs has been realized by an *Advanced Peripheral Bus* (APB), which bases on the *Advanced Microcontroller Bus Architecture* (AMBA) technology. The bus master (Spartan3A device) manages the data transfer between an onboard USB-chip device and the bus slave (Virtex 5 device) as an arbiter, cf. Fig. 4.10. A shared memory realizes the data exchange between the analyzing cryptographic implementation on the Virtex 5 and the APB slave module. The cryptographic implementation under analysis can directly access the data in the requested word width without any synchronous control element. Thereby, the core under test is solely running on the target device after the data is transferred into the shared memory. The shared memory also offers a 16 bit width status register and a command register, which are both connected to a small control automaton that starts the analyzing core. This automaton also writes the result of the core implementation back into the shared memory, so that it is accessible over USB via the APB bus master. The status register is only readable while the command register is completely accessible over USB. Due to the fully customizable automata, implemented cores with multiple functionalities are supported as well.

Host PC management On the host side drivers at the operational system level provide appropriate functions and operations to communicate with the evaluation board via the USB protocol. As depicted in Fig. 4.10, these drivers are encapsulated in Matlab scripts, which offer services to the measurement applications. Thereby the user has a transparent handling of the data exchange between the host and the analyzing core implementation. Due to the modular structure and standardized functions, the Matlab measurement application is generic and can easily be adapted to new algorithms. The USB interface to the SASEBO-GII is also utilized by a small script to reconfigure the Virtex 5 via a configuration

file from the host PC. Thus, the measurement application is able to directly reconfigure the device before the actual measurement routine starts.

Besides the reconfiguration and data exchange with the evaluation board the host PC also controls the digital oscilloscope via the measurement application. The application sets up the oscilloscope to the appropriate measurement settings before it is used in the measurement routine. The measurement routine coordinates both components, the evaluation board and the oscilloscope. Furthermore, the application is responsible to keep the set of the captured \mathcal{L} from the evaluation board and the used parameter \mathbf{x} and \mathbf{y} of each of the measurements runs consistent. This requirement is of great importance to assure an error-free processing of this data in the evaluation phase, which yields the side-channel resistance confidence of the analyzed core.

Application I: Symmetric-Key Algorithm AES

Contents

5.1	Considerations about the Architecture	95
5.1.1	Partitioning of the AES Modules	96
5.2	Implementation of the Mutating Data Path	98
5.2.1	Mutating SBoxes	100
5.2.2	Concurrent SBoxes Management	104
5.2.3	Merging Round and Routines	107
5.2.4	Side-Channel Analysis of the Data Path	109
5.3	Discussion of the Evaluation Results	111

In this Chapter the application of the concept of mutating data paths on symmetric-key based cryptographic algorithms is demonstrated. The Advanced Encryption Algorithm (AES) has been selected, because it is one of the most frequently used encryption algorithms today.

First, the general implementation structure of the example will be discussed. After the general partitioning considerations of the AES-128 architecture the countermeasure concept for the *AES Mutate* is introduced. All the proposed basic design methods, including online allocation, dynamic binding, and flexible scheduling, are executed in order to get an AES implementation with a mutating data path. The impact of manipulating the \mathfrak{dp} (processing unit, concurrency, task) of the AES Mutate aimed for decreasing the exploitable characteristic power consumption will be evaluated with the metrics introduced in Sect. 4. In the end the proposed side-channel hardened AES design is compared with an assimilable AES design with countermeasures proposed in the literature .

5.1 Considerations about the Architecture

In this chapter the presented application example, using AES, demonstrates the feasibility of applying the mutating data path concept on symmetric-key

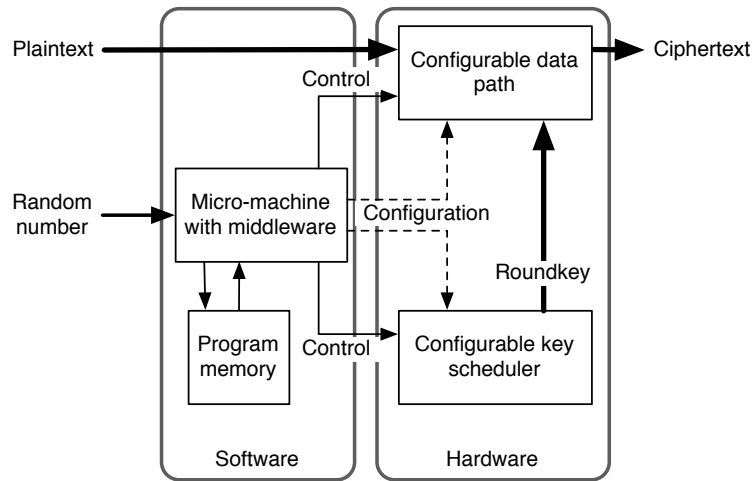


Figure 5.1: SW/HW partitions of the AES Mutate architecture

cryptographic algorithms. Due to the structure of the block cipher algorithm the attacks and the architecture of the decryption part are quite similar and thus would not gain additional insight for reasons of demonstrating the feasibility of the proposed concept of mutating data paths.

One property of mutating data paths is the manipulation of in parallel processed data, therefore the planned architecture of AES Mutate has to be able to process a 128 bit-width data word within one clock cycle. The four basic operations of the AES, which transform the data from the plaintext after 10 rounds into the ciphertext, should also be modularized into efficient hardware components in order to achieve a 128-bit data processing. Therefore, the key scheduler has to be able to provide a new complete roundkey every clock cycle.

The interface for handling the AES Mutate as well as the handling of the algorithm itself should be very similar to the handling of a non-mutating hardware AES implementation. An additional port is needed to provide a source of entropy, e.g., a random number generated by a secure random number generator, to the architecture for mutating the data path non-deterministically. The architecture should be able to process incoming encryption requests without a larger latency between the incoming plaintexts, so that the data can be streamed to the AES Mutate core.

5.1.1 Partitioning of the AES Modules

The general overview of the software/hardware partitioning of the AES mutate is given in Fig. 5.1. As depicted by the figure the major part of the software partition controls the data flow as well as the configuration of the architecture.

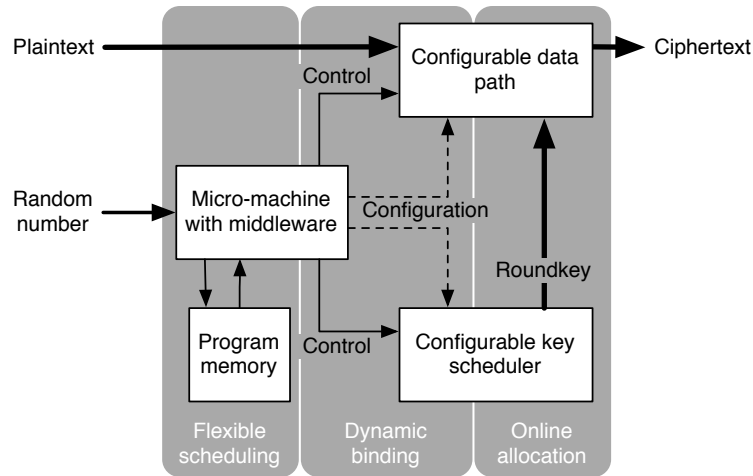


Figure 5.2: Applied methods on the AES Mutate architecture

Therefore, the behavior in terms of reconfiguring the system and executing an encryption from the controlling point of view is easy to adapt. The VLIW opcode for the configuration of the data path as well as the opcode to control the in hardware implemented basic operation for an encryption is stored inside the program memory. A BRAM module of the FPGA platform can be utilized for this purpose. A small micro-machine addresses the program memory in the correct order. The functionality of the middleware is separated into a software part, embedded into the micro-machine, and into a hardware part. The software part of the middleware manages the control flow and the execution order in case of changing the binding. The hardware part of the middleware is embedded in the data path, where it controls the data flow. The basic operations for executing a round operation are embedded in a reconfigurable data path as hardware components. Also the needed operations for expanding the secret key to the according round key are grouped to the hardware partition. The hardware area of the key scheduler is also reconfigurable, so that the variation of the round execution in terms of in parallel processing intermediate values or execution length does not affect the computational correctness of the result.

Figure 5.2 depicts the design-specific reasons for the mutating procedure to partition the functionalities in that manner. The interface between the two proposed methods flexible scheduling and online allocation is the dynamic binding layer. This layer is realized by components of the micro-machines as well as architecture entities with the data path and the key scheduler. In detail the middleware components are used to realize the dynamic binding as it is a central element according to the proposed design flow in Fig. 3.15. The other two methods are strictly partitioned into either the hardware or software domain.

5.2 Implementation of the Mutating Data Path

In this section the implementation procedure of the mutating data path is detailed. First the attack threat on AES is discussed in order to identify, which basic operation is most vulnerable due to its power consumption signature and thus has to be secured. Based on this discussion the different properties of the mutating data path are reasoned in terms of which parts of the data path have to be reconfigurable to increase the effort of a power analysis attack. The development of the data path is oriented on the design flow and will include all proposed methods of the mutating data path to increase the security of the design.

Threatening scenarios The most focused targets of power analysis attacks on block cipher are the key scheduler or the first and last round operation of the encryption process. As the key scheduler is planned to be implemented in hardware with a fixed runtime and the key expansion is done in parallel, no power analysis attacks, as reported in [Man02], can be conducted efficiently. A bigger threat therefore is the attack of the first or the last round of the AES. These two rounds are of great interest because the adversary is able to recalculate every possible intermediate value for those rounds with the knowledge of the plaintext or the ciphertext. In both attack scenarios the adversary usually tries to exploit the power consumption of a bijective operation with a strong value diffusion. Therefore the substitution functions, i.e., the SBoxes, of a block cipher algorithm, is the ideal candidate for this exploration. Of course, the processed value of the SBox has to be dependent on the secret key value of the encryption algorithm. In case the adversary can not directly exploit such a substitution function for his or her attack, he or she can try to exploit the operation, which processes the secret value directly with the current intermediate value. Usually this function is a non-bijective function of two parameters, e.g., a XOR-operation, thus the attack is not so effective, because multiple combinations of the two parameters lead to the same result and power consumption.

The scenario of attacking the final round of AES usually exploits the bit flips of the register transition from the previous intermediate value of the state to the final ciphertext value stored in the register. Thus, the leakage function of attacking the i -th subkey $k_{(i)}$ of the last round $\mathcal{L}_{\mathcal{F}_{t, \text{AES_last}, \text{CSA}^9}}(k_{(i)}, c_{(i)})$ is composed by a 9-dimensional subspace $\mathcal{F}_{9,t;k}$ and refers to the following j basis vector functions:

$$\begin{aligned} g_{0,t;k}(c_{(i)}, k_{(i)}) &= 1 \\ g_{j,t;k}(c_{(i)}, k_{(i)}) &= \left(c_{(i)} \oplus \text{Sbox}^{-1}(\text{ShiftRows}(c_{(i)} \oplus k_{(i)})) \right)_j - \frac{1}{2^1} \quad \text{for } j = 1, \dots, 8 \end{aligned} \tag{5.1}$$

where $c_{(i)}$ denotes the corresponding byte of the ciphertext, which is processed with $k_{(i)}$ in the last round. Instead of using a subspace $\mathcal{F}_{9,t;k}$ to describe the

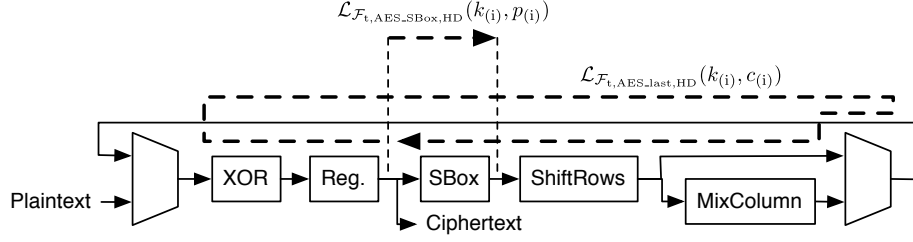


Figure 5.3: Leakage function of an exemplary AES data path

bit flips, the Hamming distance over the lower expression in Eq. 5.1 without the subtraction of $\frac{1}{2}$ can be used as well. The switching activity of the SBox in the final round of the AES is the targeting activity to be exploited, cf. the Hamming distance in Eq. 5.2.

$$\begin{aligned}\mathcal{L}_{\mathcal{F}_{t,AES_SBox,HD}}(k_{(i)}, c_{(i)}) &= \text{HD} \left(c_{(i)}, \text{Sbox}^{-1}(\text{ShiftRows}(c_{(i)} \oplus k_{(i)})) \right) \\ &= \text{HW} \left(c_{(i)} \oplus \text{Sbox}^{-1}(\text{ShiftRows}(c_{(i)} \oplus k_{(i)})) \right) \quad (5.2)\end{aligned}$$

With small changes to the Hamming distance model in Eq. 5.2 the SBox output of the first round can be attacked by the following leakage model:

$$\begin{aligned}\mathcal{L}_{\mathcal{F}_{t,AES_SBox,HD}}(k_{(i)}, p_{(i)}) &= \text{HD} \left(p_{(i)}, \text{SBox}(p_{(i)} \oplus k_{(i)}) \right) \\ &= \text{HW} \left(p_{(i)} \oplus \text{SBox}(p_{(i)} \oplus k_{(i)}) \right) \quad (5.3)\end{aligned}$$

Of course, an equivalent leakage model, $\mathcal{L}_{\mathcal{F}_{t,AES_SBox,CSA^9}}(k_{(i)}, p_{(i)})$, with a subspace, $\mathcal{F}_{9,t;k}$, can also be used for attacking the SBox of the first round:

$$\begin{aligned}g_{0,t;k}(p_{(i)}, k_{(i)}) &= 1 \quad (5.4) \\ g_{j,t;k}(p_{(i)}, k_{(i)}) &= \left(p_{(i)} \oplus \text{SBox}(p_{(i)} \oplus k_{(i)}) \right)_j - \frac{1}{2^1} \quad \text{for } j = 1, \dots, 8\end{aligned}$$

In order to exploit a register transition of the first round either the previously stored register value has to be determined or the next register value has to be calculated. For hardware implementations it is possible that the next register value after the first round is the complete second round including the Mix Column function, which puts the intermediate calculation at a not affordable cost. The effort of calculating the intermediate values of a targeting byte rises in that case from 2^8 to 2^{40} . Figure 5.3 visualizes the described leakage on an example data path. The attack scenarios of both leakage functions are depicted on the schematic of an AES data path.

Properties of the mutating data path Based on the threat scenarios the mutating data path should have the following properties:

- Executing the SBox operation on several different implementations
- Dynamically changing the degree of in parallel processing SBox units
- Merging round operations into one clock cycle
- Manipulating the word-width that is processed during a clock cycle
- Randomizing the state representation in the shared memory

The SBox operation is selected for the online allocation as well as for the dynamic binding, because it is the most vulnerable operation in two of three different round operations. In two of the three presented exemplary attack scenarios the SBox operation was a central element to determine the estimated exportable power consumption of the block cipher. It is a bijective operation with a high amount of diffusion and, as reported in [MS02], it has the highest power consumption of all the different basic operations of the AES. Also there are a lot of different kinds of implementations of SBox processing units reported in the literature with different design specific properties, cf. [SMTM01, MS02]. This design specific properties are exploited to manipulate the characteristic power consumption of the operation processing the same data.

Due to the higher power consumption of in parallel processing SBox implementations the variance of the power consumption signature will get broader. Thus, the SNR of extractable information within the captured power traces will decrease and thereby increase the attacking effort. The varying of the operation per clock cycle as well as merging round operations (routines of the AES, cf. Fig. 3.13) are used in order to hide the number of concurrently active processing SBoxes. In order to apply flexible scheduling this property can be seen as a mixture of shuffling [MPO07] and the temporal jitter countermeasure [MGV08]. The last action point of the previously listed properties shall prevent such an exploitation of the XOR operation by a sort of randomized register pre-charging and shuffling. Please note that the design specific properties of the mutating data path are mentioned in Subsect. 5.1 and have also be considered.

5.2.1 Mutating SBoxes

In this Subsect. the different SBox implementations for the online allocation are described. First, the structure and design properties of each utilized SBox type is introduced, focusing on the designing properties from a CAD point of view. Therefore, the amount of resources are compared as well as the timing properties. After that the different implementations are evaluated according to their common power consumption signature. Therefore, the introduced method in Subsect. 3.2.2 is used to compare the different designs. The comparison is

conducted on real measurements and not simulated power traces in order to analyze the behavior of the implementations and the interference on the power consumption in practice. The measured power consumption is derived from a voltage drop over a measurement shunt and the value of the shunt, cf. Fig. 4.9.

5.2.1.1 Implementation of Different SBoxes

Due to the flexibility of the mutating data path, the design requirements and the runtime properties of the AES Mutate, all SBox implementations have to be implemented by combinatorial logic only. So, in terms of using the Virtex 5 as FPGA platform, the SBox implementations are constrained to only utilize LUT and MUX primitives of the FPGA architecture, as well as the interconnection network for linking the different basic primitives. Therefore, no specialized macro cell modules of the Virtex 5 platform, e.g., BRAM cells, are used.

As foundation for the different SBox implementations two different basic approaches of implementing a SBox are selected. The first basic approach is a lookup-table based SBox implementation, where the mapping of the SBox is done directly by a large value array. The second approach is to exploit the mathematic nature of the irreducible polynomial in $GF(2^8)$ by tower field constructions or by so-called composite field implementations.

SBox implementations Six different SBox implementations were selected for the online allocation method of the mutating data path. All analyzed implementations are embodied based on a description of the given reference in the literature. The various SBox implementations are listed in Tab. 5.1 together with their design properties. The two main design properties of the FPGA platform, occupation of LUT and the delay of the critical path, are also important selection criteria of the SBox types of the mutating data path in order to meet the design requirements.

The first SBox implementation, TBL, is a look-up table based implementation. The second design PPRM1 formulates the input and output mapping of the non-linear substitution function as a boolean expression in the Reed-Mueller notation¹ for each bit line. The design PPRM3 utilizes the approach of a composite field based SBox and partitions the SBox design into three sections and describes the functionality again in the Reed-Mueller notation. The pre-inversion section (isomorphic mapping and scaling to subfield), the inversion, and the post-inversion section (inverse isomorphic mapping and affine transformation) are individually expressed by a Reed-Mueller notation based equation.

¹The functional behavior is constructed only by the boolean functions featuring XOR and AND operations only

Table 5.1: Resource consumption of the different SBox implementations

SBox design	Based on	Type	LUT	Critical path delay
TBL	[PP10]	Lookup-table	32	4.5 ns
PPRM1	[MS02]	Boolean expression	80	5.6 ns
PPRM3	[MS02]	Composite field	64	8.5 ns
COMP	[SMTM01]	Composite field	50	9.2 ns
COMP2	[Koc08]	Composite field	63	9.7 ns
HYP1	[Koc08]	Lookup+Composite	55	10.4 ns

The COMP design is partitioned into five sections, but using the same subfield composition as PPRM3. The isomorphic mapping is not merged with the subfield scaling in one boolean expression and also the inverter is separated into two sections, which are expressed by the boolean operations AND and XOR. The design COMP2 uses the same approach as PPRM3 and COMP by representing the different sections of the design in the Reed-Mueller notation but utilizing another combination of subfields. The design HYP1 is a modified form of the COMP2 implementation: the inversion module in $GF(4)$ is not constructed by a boolean expression. Instead, it is modeled by a lookup-table.

Due to the requests of the design properties only the four fastest design are considered for the evaluation of their characteristic power consumption signature.

Evaluation of the different types For the evaluation of the data-dependent power consumption signature the total correlation is used. A set of 50,000 power traces of each SBox implementation was captured, mentioning that the different designs were stimulated with the same input data during the capture process. The total correlation evaluation compares the power consumption distribution of Design A of each Hamming distance (HD) class with the power consumption distribution of design B of each HD class, cf. Subsect. 3.2.2.

Figure 5.4 visualizes the result of the evaluation of the SBox implementations against each other. If the total correlation is 1 the distributions are identical, if the value is 0 they are completely uncorrelated. The black colored frames indicate, which HD class of design B fits best to a HD class of design A due to the correlation of their power consumption distribution of the HD classes. It is evident that in most cases the mapping of the power consumption distribution of one HD class of design A does not most likely fit to the distribution of the same HD class of the comparing design B. For instance, in Fig. 5.4(a) the power consumption distribution of HD class 0 of design A (PPRM1) is likely more similar to the HD class 4 of design B (PPRM3). Figure 5.4(a) also displays another beneficial feature of using the both designs together for the online allocation.

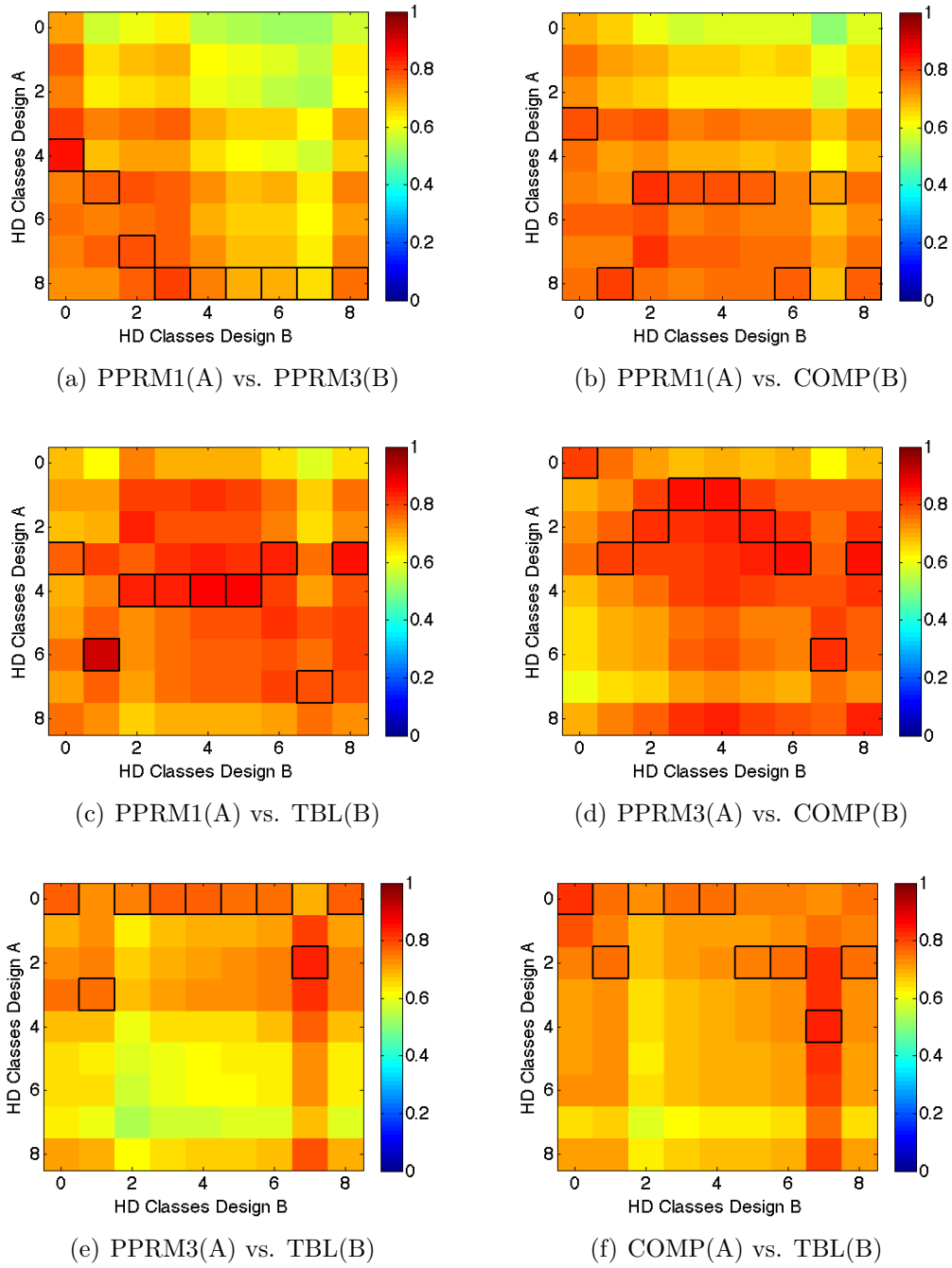


Figure 5.4: Evaluation of the SBox implementations by the total correlation over the different Hamming distance classes

Multiple power consumption distributions of HD classes of PPRM3 are mapped to the power consumption distribution of HD class 8 of the PPRM1 design.

A similar kind of information compression² can be observed in four other evaluation cases. The evaluation further reveals that the weakest combinations of designs for the online allocation are the combination of the SBox designs PPRM3 and COMP. For both designs the power consumption distribution of the HD class 0 is most likely and no other HD classes are mapped to the HD class 0. Therefore, the power consumption of a Hamming distance of 0 is nearly similar for both designs, and thus does not disturb the power consumption by changing from a PPRM3 design to a COMP design in this special case. This issue can be addressed by the dynamic binding and flexible scheduling method in order to vary the amount of in parallel processing operations in order to envelop the exploitable power consumption in noise.

5.2.2 Concurrent SBoxes Management

Due to the structure of AES four different types of SBox implementations should always be used to process the intermediate data of the current AES round. In order to save space on the FPGA platform the following concurrent SBox management is used to change the SBox type of each byte section of the AES. The previously introduced virtualization method is used to change the binding between the state register and the SBox implementations in addition to the partial reconfiguration. Not every SBox in the data path is partially reconfigurable by the reconfiguration design flow, as depicted in Fig. 3.15 in order to save resources and execution time.

Figure 5.5 provides a general overview over the structure and architecture of the mutating data path. As mentioned before the data path for the transformation of the plaintext to the ciphertext (upper circuit part) as well as the data path for the key expansion (lower circuit part) have to be mutable. The key scheduler circuit is extended with a multiplexer in order switch between the updated bytes of the round key value or the previous bytes of the round key values. Thus, the key scheduler can process the next round key at once 128 bit in parallel or in 96 bit, 64 bit, or 32 bit wide chunks.

The mutating data path is able to perform with the same granularity of the in parallel processed bits. Furthermore, the degree of in parallel processed data is scalable in 8 bit steps, reaching from 8 bit up to 128 bit per clock cycle. The challenge of changing the degree of in parallel processed data through the SBox processing unit stems from the entangled data dependency between the columns and the rows of the AES round state matrix. The P-units work on the columns of the matrix and they are interconnected by a reconfigurable ShiftRows function (*RSR*), cf. Fig. 5.6(b) in order to process the complete state matrix. The RSR

²The used side-channel distinguisher $\mathcal{D}(\mathcal{L}_t, \cdot)$ maps the power consumption value of \mathcal{L}_t to the wrong HD class.

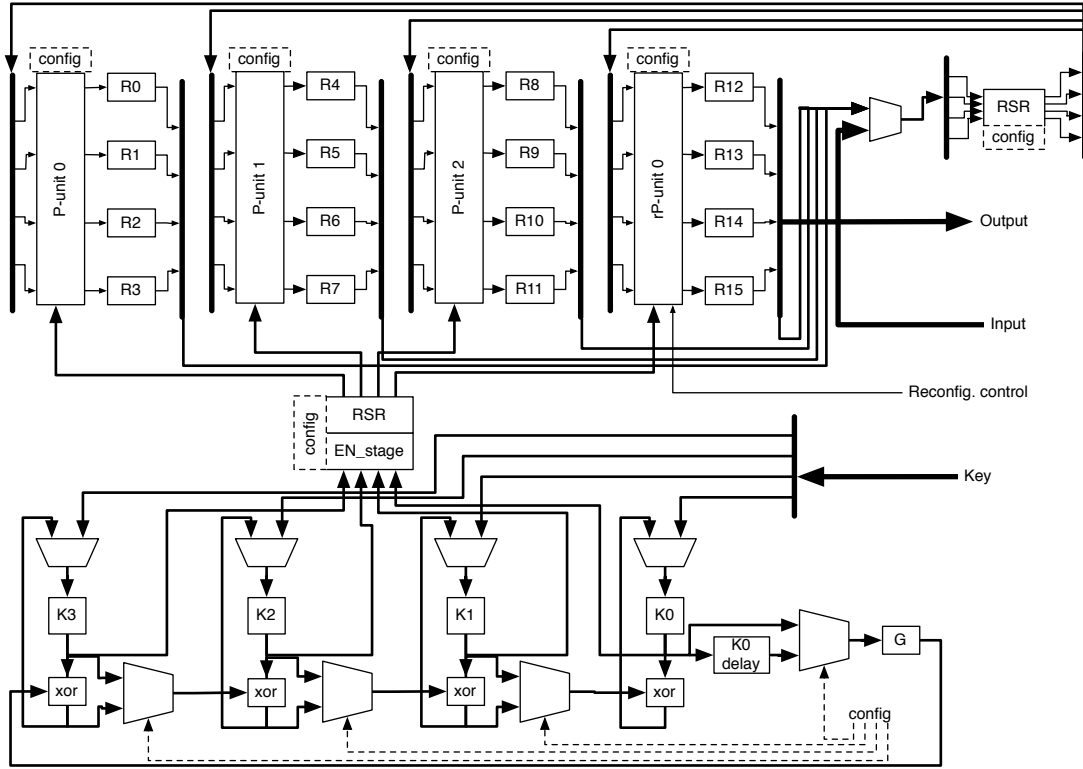


Figure 5.5: Scheme of the mutating data path

can shift each byte of a rows from one byte to the right up to two bytes on the left, as well as bypass the shift operation in order to maintain the word position of the 4 bytes on the rows. This unit can also be used to either shuffle the state representation on-the-fly as well as to change the binding to a specific SBox implementation. In case of shuffling the state, the subkeys have also to be shuffled in the same order, thus an additional RSR unit is attached between the P-units and the key scheduler. Therefore, the state randomization can be inserted within a round operation without needing additional clock cycles for the reconstruction of the original value of the state register position.

The SBox is embedded in the so-called P-unit, which also offers the functionality of the AES specific basic operations MixColumn and AddRoundkey, as well as a bypass functionality. The AddRoundkey function is merged with the MixColumn function to the so-called *MXOR*, which is configurable to perform the MixColumn operation following by an XOR operation for the AddRoundkey functionality or only to perform the XOR operation. Thus, the P-unit provides three functionalities, which can be performed combined at any level or be exe-

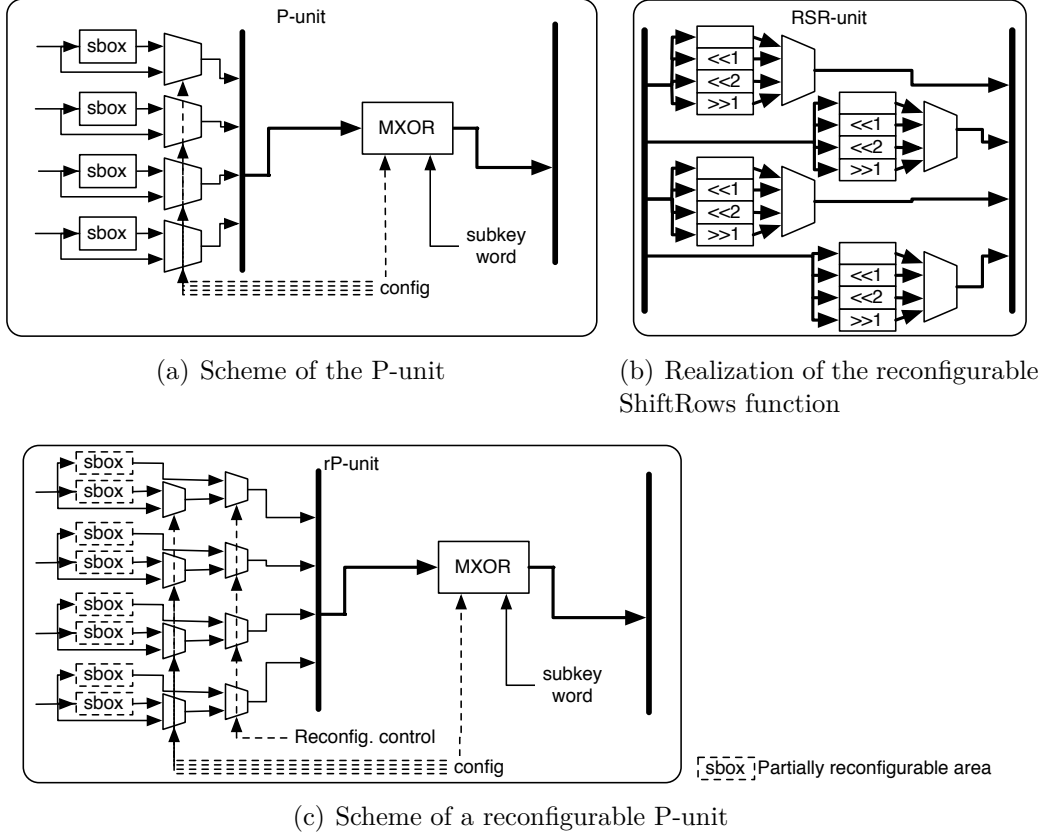


Figure 5.6: Hardware realizations of the Basic AES operations

cuted individually³ and a bypass functionality, cf. Fig 5.6(a).

In order to support the partial reconfiguration on the Virtex 5 platform at moderate costs in terms of resources and execution time, a so-called reconfigurable P-unit (*rP-unit*) is added to the data path instead of a forth P-unit, cf. Fig 5.5. The *rP-unit* contains an additional layer of multiplexers to route the data flow between the partial reconfigurable areas for the SBox implementations, cf. Fig 5.6(c). While one reconfigurable area is updated with a new configuration the other area can perform the SBox operation.

Virtualization Scheme As mentioned in Subsect. 5.1.1, the data flow routing functionality of the middleware is embedded in the discussed data path. The virtualization in this design can be conducted by utilizing the RSR units to reroute or shuffle the content of the register states. Thereby, different SBox implementations are utilized for each byte-width entry of the AES state matrix, if each P-unit/*rP-unit* instance has different SBox implementations. The config-

³SBox, MixColumn, AddRoundkey

uration settings of the P-unit/rP-unit are the other control element supporting the virtualization in order to realize the method of dynamic binding. Due to these settings the amount of in parallel working SBoxes per clock cycle can be controlled.

The software side of the virtualization can be realized by a VLIW opcode. The VLIW architecture is clearly integrated in the proposed data path of AES Mutate as well as the VLIW opcode commands that are controlling the configuration settings of the P-units/rP-unit and the RSR units. Table 5.2 provides an overview of the command schemes of the AES Mutate VLIW opcode structure. Due to the flexible VLIW architecture and the hardware architecture of the embedded middleware functionality into the data path the virtualization is a suitable interface between dynamic binding and flexible scheduling. The configuration settings P-unit/rP-unit can also be used to change the amount of in parallel working hardware components. For instance, the execution of multiple XOR operations can be manipulated in order to change the degree of in parallel working XOR operations, as well as the amount of basic AES operations executed within one clock cycle.

5.2.3 Merging Round and Routines

The last proposed method of realizing a mutating data path, i.e., flexible scheduling, is not only applicable on the XOR-operation. Due to the modularized, flexible, and reconfigurable HW/SW-codesign architecture of the AES Mutate the execution of every basic AES operation can be randomized. The application of flexible scheduling offers the ability to change the degree of in parallel working processing units of all basic AES operations in this data path. The processing word width of the data path is adaptable for all operations from 32 bit over 64 bit and 96 bit up to 128 bit within one clock cycle. The HW/SW-codesign of the micro-machine in combination with the embedded middleware infrastructure inside the data path provides a flexible rescheduling via the VLIW operations, independently of the current data path processing width. Therefore, the number of different data-dependent operations processed within one clock cycle is randomizable and thus, basic operations of different AES round operations can be merged.

5.2.3.1 Flexible Micro-Programs

The flexible rescheduling of the AES encryption is possible by utilizing the VLIW opcode commands in small micro-programs. Each AES round operation or merged round operations are composed of such micro-programs and consist of at least one micro-program. The micro-program is executed by the small micro-

Table 5.2: VLIW structure of AES with mutating data path

Bit	35	34	33	32	31:24	23:16	15:12	11:8	7:0
Function	last	rEN	loD	loK	RSRdata	confP	newRK	prKey	RSRkey

Function	Description
last	Indicates the last operation of a round.
rEN	Enables the registers to store the current intermediate values if they are set to logical one.
loD	Enables the data path to load a new plaintext for encrypting it.
loK	Enables the key scheduler to load a new key value, which is then used to derive the round key values.
RSRdata	Configures the ShiftRows function in the data path.
confPU	Configures the functionality of the P-unit.
newRK	Indicates to the key scheduler that a new round key is needed.
prKey	Enables the processing of the current round key in the data path.
RSRkey	Configures the ShiftRows function for the key scheduler to the data path.

machine attached to the data path. In every clock cycle the program counter of the micro program is incremented and executes one VLIW opcode command. In this clock cycle the opcode-command controls the processing units of the basic AES operations and sets the configuration of the data path, thus, the word width of the data path is defined by the opcode command.

Table 5.2 depicts the VLIW structure of the AES Mutate and details the VLIW opcode. One opcode command is 36 bit wide and thus is perfectly storable in a BRAM module of the Virtex 5 platform. The first four most significant bits⁴ of the opcode are control signals, which do not reconfigure the data path of the AES Mutate. The MSB bit, bit 35, is used as a delimiter in order to identify the end of a round operation. This delimiter alerts the micro-machine to load the next micro-program by jumping to a randomly selected start address of a suitable micro-program instead to increment the program counter. Therefore, the complete AES encryption can be composed by randomly selecting micro-programs for each round operation of the AES algorithm.

⁴Little endian is used and thus the MSB is on the left side

The lower 32 bits are used to reconfigure the data path during runtime in order to randomize the power consumption. The micro-programs are therefore more or less reconfiguration settings to change the architecture of the data path dynamically. This context based reconfiguration approach was selected, because the partial reconfiguration done by the Xilinx Inc. design flow takes too long. Reconfiguring a P-unit by this procedure will take at least 25 msec, in that amount of time the design can perform 25 encryptions. In order to hide the reconfiguration time all P-units have to be replaced by rP-units, which will significantly increase both, the resource consumption and the minimal delay of the critical path. Therefore, the tradeoff with only one rP-unit and a flexible micro-program structure has been selected. The depicted AES Mutate design in Fig. 5.5 contains three P-units and one rP-unit. This design is also used for the side-channel evaluation of the AES Mutate.

For instance, the opcode of a micro-program for executing an intermediate AES round within one clock cycle is *0xC1BFFFF0* with or without partial reconfiguring the rP-unit in parallel. It is also possible to compose micro-programs for dummy operations or to perform the shuffling of the register states in an individual clock cycle, which both will affect the throughput performance.

5.2.4 Side-Channel Analysis of the Data Path

The SNR metric, introduced in Subsect. 4.2.2, is used for the side-channel evaluation of the AES mutate in order to determine its vulnerability to power analysis attacks. All three already described threat scenarios are considered during the analysis of the AES Mutate. In order to visualize the increased effort, an unsecured AES design is evaluated with the same threat scenarios. The unsecured implementation is a fully parallel implementation using the COMP SBox design and will from now on be denoted to as AES COMP. The data path width of the design is 128 bit so that it can also process 128 bits per clock cycle.

The evaluation is conducted with 500,000 power traces, which are captured from the SASEBO-GII platform. The measurement procedure, where and how the power traces are captured, as well as the acquisition framework is detailed in Sect. 4.3. In case of the analysis of the AES Mutate design an additionally implemented trivium stream cipher [PP10] is used to provide the required random input for the unpredictable data path mutation. For both designs the SNR metric is evaluated for the time interval of interest. By comparing the SNR values of the COMP and Mutate designs, the increased attack effort is derivable due to the relationship of the SNR value of both evaluations, cf. Eq. (4.20). The AES COMP design is similar to the design of the DPA V2 contest [DPA] and, thus, a good reference point in order to estimate the increased resistance to power analysis attacks due to the application of the mutating data path concept.

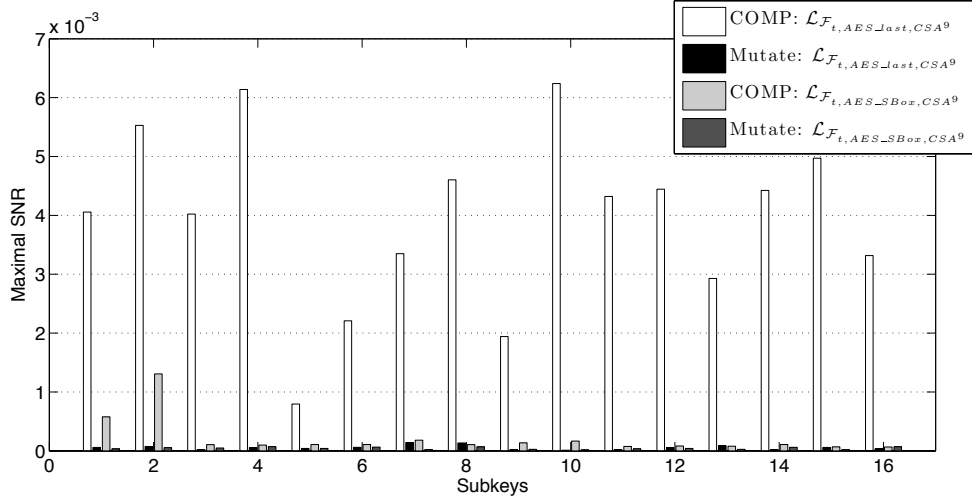


Figure 5.7: Results of the SNR metric analysis of AES Comp and AES Mutate

Figure 5.7 depicts the result of the evaluation. It displays the SNR metric values for attacking the AES COMP and AES Mutate with the leakage functions of the two previously introduced attack scenarios. The corresponding subspace descriptions for the leakage function are given in in Eq. (5.1) and Eq. (5.4). During the measurements all proposed methods for mutating the data path have been activated⁵. The attack on the reference design COMP is only successful for the first attack scenario with $\mathcal{L}_{F_{t,AES_last}, CSA^9}(k_{(i)}, c_{(i)})$, while the attack on the SBox with $\mathcal{L}_{F_{t,AES_SBox}, CSA^9}(k_{(i)}, p_{(i)})$ is not successful. Comparing the SNR values of the attack on the last round of both designs, the SNR level of the AES Mutate is at least nineteen times smaller as the SNR level of the AES COMP. Table 5.3

Table 5.3: SNR levels of the different subkeys

Subkey	1	2	3	4	5	6	7	8
$\frac{SNR(COMP)}{SNR(Mutate)}$	70.97	75.16	192.98	114.04	19.32	36.86	23.66	34.51
Subkey	9	10	11	12	13	14	15	16
$\frac{SNR(COMP)}{SNR(Mutate)}$	87.35	604.01	192.95	81.66	32.58	235.28	85.74	101.65

provides the ratio between the SNR levels of both designs for the leakage function focusing on the last round.

⁵Online allocation by remapping the SBox types by virtualization as well as partial reconfiguration of the rP-unit, dynamic binding and flexible scheduling using the VLIW architecture together with the micro-program approach

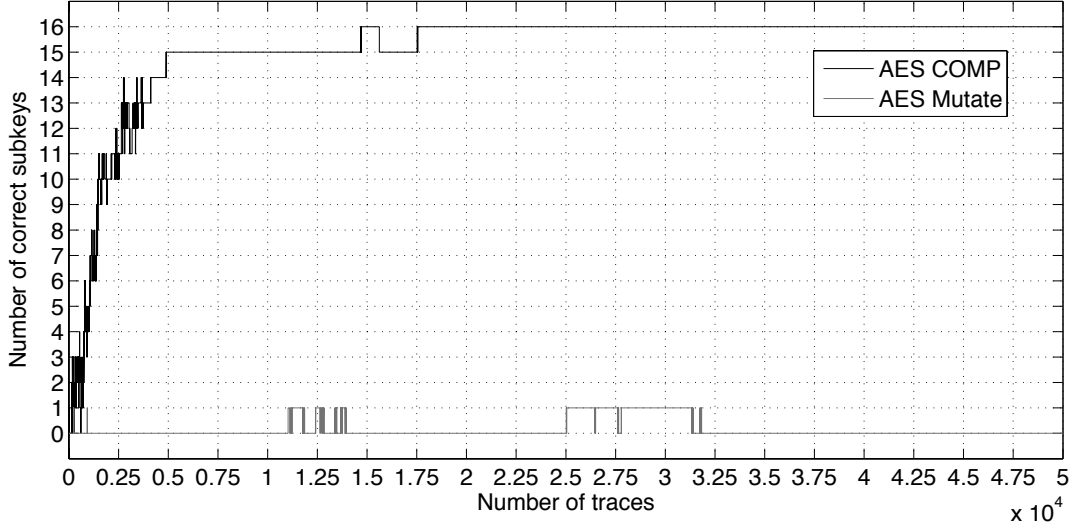


Figure 5.8: Analysis of the correctly found subkeys over number of traces

None of the subkey values of AES Mutate could be revealed by using 450,000 traces for the linear regression in the profiling phase and 50,000 traces for the attack phase. In case of analyzing the reference design, AES COMP, with 450,000 traces in the profiling phase and 20,000 traces in the attack phase, all subkey values can be revealed by using $\mathcal{L}_{\mathcal{F}_{t, \text{AES_last}, \text{CSA}^9}}(k_{(i)}, c_{(i)})$.

The history of the correctly found subkeys was analyzed in order to identify the needed number of traces to successfully attack the AES COMP design using $\mathcal{L}_{\mathcal{F}_{t, \text{AES_last}, \text{CSA}^9}}(k_{(i)}, c_{(i)})$. Thus, the attack result of the third phase of the stochastic approach for both designs was evaluated and recorded after every 100 steps. The result of this analysis is depicted in Fig. 5.8. All subkeys of the AES COMP can be revealed after using 17780 traces. After this amount of traces the result is stable, while the subkey values of the AES Mutate design seems to not converge to the correct value of any subkey. As visualized in Fig. 5.8 the analysis does find one subkey value at maximum after a certain number of traces, but then chose a false subkey value.

5.3 Discussion of the Evaluation Results

The inhere discussed countermeasure for sure belongs to the class of hiding based countermeasures. Well-known from the literature, hiding countermeasures "only" increase the required effort of the attack by increasing the noise compared to masking based attacks, which scramble the intermediate value to secure the com-

Table 5.4: Resource consumption of various AES designs

Design	Ref.	LUT	Reg.	BRAM	Freq. [Mhz]	Throughp. [$\frac{\text{Bits}}{\text{cycle}}$]	Cycles for enc.	Platform
AES-128 without countermeasures								
TBL		1225	276	0	284	128	11	Virtex 5
PPRM1		1881	276	0	264	128	11	Virtex 5
PPRM3		2028	276	0	187	128	11	Virtex 5
COMP		1726	276	0	164	128	11	Virtex 5
AES-128 with countermeasures								
Mutate		2957	387	0	101	32-128	14-51	Virtex 5
Boolean masking	[RYS11]	4772	904	0	100	128	11	Virtex 5
Multiplicative masking	[OO07]	7628	1291	0	44	128	-	Virtex-E
Threshold	[HF12]	12627	1438	0	79	128	-	Virtex 5
SDDL	[SVKH10]	1410	518	20	37	128	-	Spartan3
WDDL	[HF12]	7292	1160	0	56	128	-	Virtex 5
MDPL	[HF12]	12140	1205	0	44	128	-	Virtex 5
Intra-masking	[HF12]	5898	546	0	141	128	-	Virtex 5

putation. But considering second order attacks masking also "only" increases the required effort of the attack. A classical second order attack like in the masking case is not directly applicable for this countermeasure, because the computational secret is not shared. Nevertheless, it may be possible to conduct an attack on the micro-machine in order to gain the executed opcode of the VLIW architecture. But due to the parallel processing and the balancing of the addressing scheme for the micro-programs, it seems more promising to sort and cluster the different power traces based on the power consumption signature. The development of an efficient sorting scheme to cluster the different traces and identifying them is one open question for future research of this thesis. Due to the combined hiding methods based on online allocation, dynamic binding and flexible scheduling the normal approach to fight shuffling or dummy operation countermeasures with the known method of windowing is not directly applicable. The changing amount of operations per clock cycle in terms of types of operations and degree of in parallel processed bits makes this new countermeasure not directly comparable to shuffling or dummy operations.

Another criterion to rate the efficiency of a countermeasure is the additional resource consumption in order to harden the design. Table 5.4 provides an overview of the resource consumption of designs with and without countermeasures. The unsecured implementations are realized by utilizing the same types of SBox designs that are used for the online allocation. Thus, the unsecured designs provide a good comparison of the additional resource consumption of the AES Mutate. The table also lists other countermeasures reported in literature

of both countermeasure classes⁶, which are applied on the Virtex 5 architecture or at least implemented on a Xilinx FPGA. The discussed countermeasure AES Mutate is followed by three masking countermeasures, then by two hiding countermeasures, and in the end two 1 bit masking schemes are listed. One advantage of the here presented AES Mutated is that the resource consumption is only nearly doubled, while the speed is only halved. In terms of the least speed lost the AES Mutate is the second best design, while in terms of the least additional resource consumption it is the design with the lowest used amount of LUTs, registers, and BRAMs. Note that the mutating data path design is still combinable with a classical masking scheme, for instance boolean masking. Mutating data path seems to be an efficient hiding implementation method for this block cipher.

The cost for the improvement against power analysis attacks of the AES Mutate is the longer execution time and thus a drop down in throughput. The amount of required clock cycles for encrypting a plaintext may vary from 14 to 51 clock cycles, because it depends on a random number. Nevertheless, the decreasing throughput is moderate compared to other countermeasure proposals listed in Tab. 5.4. So, this countermeasure is very appropriate for embedded systems utilizing a small FPGA module, which do not want to spend much additional resources and still want to achieve a moderate computational speed.

⁶Hiding and Masking

Application II: Public-Key Algorithm ECC

Contents

6.1	Hardware Architecture Considerations	116
6.1.1	Partitioning of the ECC Coprocessor	117
6.2	Realization of the Mutating Data Path	118
6.2.1	Mutating Finite Field Multipliers	121
6.2.2	Management of Concurrent Multiplication Units	125
6.2.3	Various Projective Coordinates	128
6.2.4	Side-Channel Analysis of the Data Path	130
6.3	Discussion of the Evaluation Results	133

Next to the symmetric-key cryptography the asymmetric-key cryptography is an important field of algorithms in order to secure data, communication, or systems. The application of the mutating data path concept on an asymmetric-key cryptography algorithm is demonstrated in this chapter. The elliptic curve cryptography algorithm (ECC) has been chosen for these studies, as it has been in the practical field for several years and is used in many systems for electronic signature schemes or key-exchanges. An ECC-coprocessor has been designed in order to accelerate the central operation of this asymmetric cryptographic algorithm with a hardware based scalar multiplication.

A general overview of the design and architecture of the coprocessor will be provided to the reader at first. In the following section after the partitioning of the architecture of the ECC-coprocessor design the countermeasure concept of the inhere introduced *ECC Mutate* designs are discussed. This discussion also includes several threat scenarios of side-channel attacks to the ECC algorithm. Based on these scenarios the properties $\mathfrak{dp}(\text{processing unit, concurrency, task})$ will be adapted by all the introduced proposed methods of the mutating data path¹.

¹Online allocation, dynamic binding, and flexible scheduling

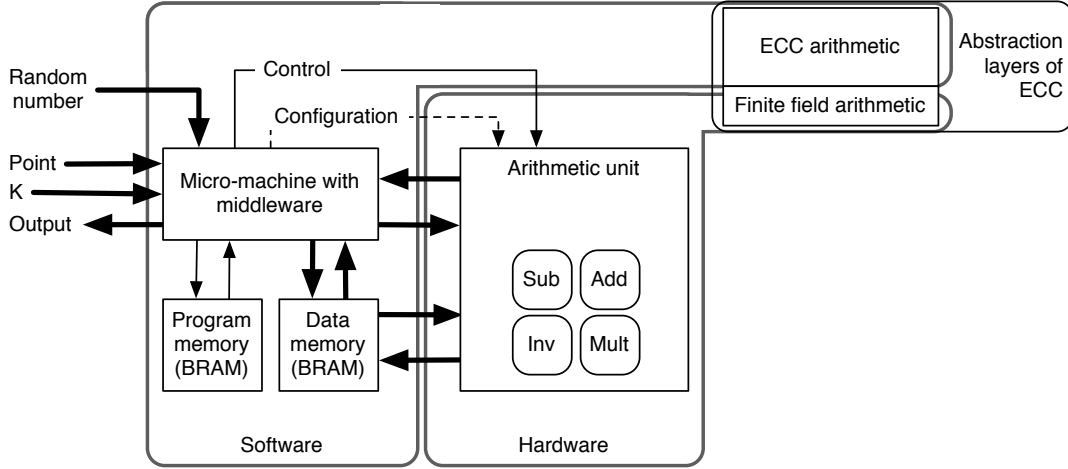


Figure 6.1: SW/HW partitions of the ECC Mutate architecture

The impact on the varying $\mathfrak{dp}(\text{processing unit}, \text{concurrency}, \text{task})$ due to the applied methods of the mutating data path like dynamic binding is evaluated in terms of decreasing the exploitable power consumption signature. The side-channel evaluation is done by the metrics discussed in Sect. 4. After the side-channel evaluation the results are discussed by comparing the introduced ECC Mutate design to assimilable ECC designs and published countermeasures found in literature.

6.1 Hardware Architecture Considerations

The application example of applying the concept of mutating data path on the ECC algorithm focuses on the point multiplication operation of the elliptic curve arithmetic. Therefore, the main task of the coprocessor ECC Mutate is to perform the point multiplication on a curve in $\text{GF}(p)$ by utilizing a mutable data path architecture. The coprocessor should support the point multiplication on arbitrary prime field based elliptic curve with a bit length of up to 384 bits. In specific it should be able to process the recommended curves of the NIST institute: P192, P224, P256, and P384 by NIST, cf. [GFD09].

A hardware acceleration should be considered to achieve the appropriate computational throughput even for the point multiplication on the P384 curve. The manipulation of the second design property by dynamic binding is done on the finite field operational level. Therefore, an appropriate number of redundant hardware operation units is needed to achieve a suitable degree for the manipulation of concurrent processing units.

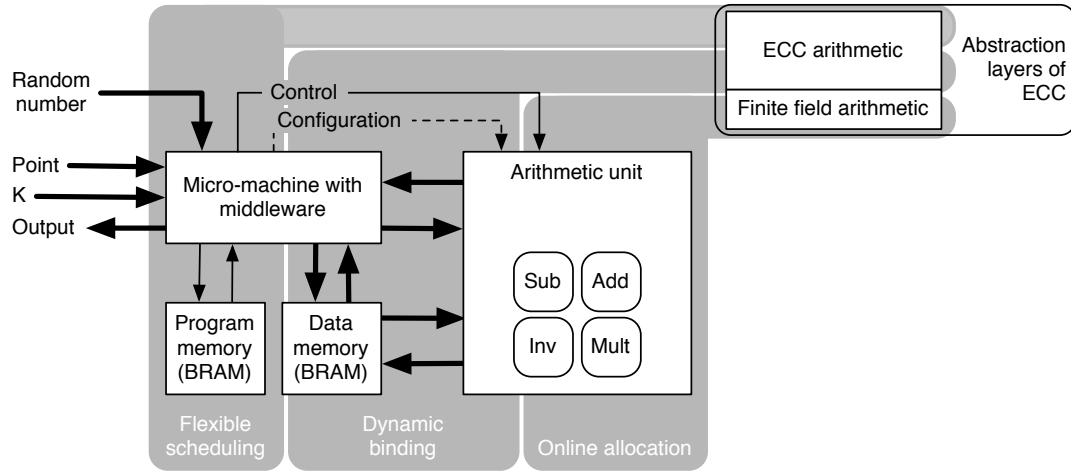


Figure 6.2: Applied methods on the ECC Mutate architecture

The ECC-coprocessor should provide a simple interface, which does not distinguish too much from a non-mutating implementation, thus ports for the key data, the basis point, and the multiplication result should be present. An additional port for the mutating version is needed in order to provide a random number from a secure random number generator to the coprocessor. This random input is used as entropy source in order to mutate the data path architecture in a non-deterministic way.

6.1.1 Partitioning of the ECC Coprocessor

The architecture of the ECC-coprocessor is based on a hardware/software code-sign. The foundation of the design is based on the results of a diploma thesis [Kus10] and a bachelor thesis [Ema11] investigating an ECC implementation on the Virtex 5 platform. The implementation of the ECC-coprocessor architecture of the design for further investigations in this thesis was done in a student project [Ema12]. Figure 6.1 depicts the partitioning of this architecture in a software and hardware section. The functionality of the finite field arithmetic is mapped to the hardware section in order to accelerate the computation of the operations: Multiplication, addition, subtraction, and inversion are the basic finite field operations, which are provided as related processing hardware units. Thus, the ECC-arithmetic operations are also improved, because the finite field processing units are utilized to compose the different operations of the ECC-arithmetic. The software of this heterogeneous architecture controls the data flow and the execution of the ECC-arithmetic operations as well as the configuration of the architecture. The multiple necessary operation and control sequences to perform

a scalar multiplication on an elliptic curve are grouped in the ECC-arithmetic layer, which is implemented in software and is executed on a small micro-machine. The software procedures, describing the ECC-arithmetic operation, are flexible to adapt in order to use different mathematical procedures or algorithms to compose a scalar multiplication with basic finite field operations. A VILW opcode infrastructure is utilized to provide a virtualizeable middleware architecture in the hardware section of the data path to achieve a higher degree of freedom for a flexible instruction handling and a high hardware reuse on this architecture.

The software for controlling the sequence of basic finite field operations, as well as the configuration of the data path, is stored inside the instruction memory of this architecture. The intermediate values and the final computational results are stored in an extra data memory. Every basic finite field hardware module is able to access the data memory, so that it can be used as a shared memory storage between the processing hardware units. For both memory modules, for storing instructions and data, the BRAM primitives of the Virtex 5 platform have been utilized.

The partitioning for the different sections of application of the three proposed methods for mutating data paths are depicted in Fig. 6.2. The method online allocation is applied on the finite field basic operations in order to manipulate the first of the design properties $\mathfrak{dp}(\text{processing unit, concurrency, task})$. The other two methods, dynamic binding and flexible scheduling, are applied on the ECC-arithmetic layer and thus manipulate the behavior of the architecture by varying the control flow of the composed scalar multiplication.

6.2 Realization of the Mutating Data Path

After the general overview of the architecture of the coprocessor ECC Mutate has been provided, the implementation of the mutating data path will be discussed in the following. Before the mutating properties of the constructing mutable data path are discussed, different threat scenarios on the coprocessor are investigated. In specific, the vulnerability of different operational steps due to the corresponding power consumption signature is discussed and thus certain attack scenarios are identified. With the result of this analysis the different properties for the mutating data path are reasoned in order to increase their resistance to power analysis attacks. The transformation of the data path into a mutating data path is oriented on the design flow, introduced in Sect. 3.5, and includes the different proposed mutating design methods.

Threatening scenarios The scalar value n of a scalar multiplication $[n]\mathbb{P}$ with a basis point \mathbb{P} on the elliptic curve E is the secret an attacker is interested to

reveal. With the knowledge of the secret scalar value an adversary may be able to break the security or the trustworthiness of an ECC based cryptographic system. The secret, i.e., the scalar value or the key (from a cryptographic point of view), controls the composition of the scalar multiplication, which is based on the ECC-arithmetic operations point-doubling and point-addition, cf. [HMOV03]. Thus, each combination of \mathbb{P} and n refers to a certain point on the curve E . An adversary may exploit the side-channel characteristic of the point-doubling and point-addition operation in order to reveal the composition of the scalar multiplication and thus the secret key.

The two ECC-operations point-doubling and point-addition are arithmetically different due to the mathematical nature of the elliptic curve in $\text{GF}(p)$ and thus have a different sequence and amount of basic finite field operations to be performed. Therefore, the first threat scenario is to identify two different patterns in the captured power traces pointing to the scalar multiplication composition and, thus, to extract the secret value of n . In some straight-forward implementations only one captured power trace is needed in order to perform a successful SPA to extract the secret value from the exploitable power consumption signature.

In case of a balanced implementation of the point-doubling and point-addition operations the attack of the first threat scenario will not work. The SPA attack will also be not successful in case that the sequence of the executed ECC-operations is identical for every value of n . The Montgomery ladder algorithm provides such properties for the scalar multiplication, cf. [CF05]. Another attack vector to overcome these circumstances is to attack the register transitions of the intermediate values after the a point-doubling or point-addition is done. To be more precise, the intermediate values of the base point transformation are attacked. Due to the algorithmic structure of the Montgomery ladder either a point-doubling operation or a point-addition operation is performed on specific intermediate values. This circumstance leads to two different possible intermediate values, which are determinable with the knowledge of the previous intermediate values or previous point on the curve E . Due to the fact that the basis point \mathbb{P} is public, the first intermediate value depending on the most significant bit of n is therefore attackable. Thus, the adversary can mount a DPA-like attack to verify his binary hypothesis, if a point-doubling or a point-addition operation is performed, leading to a specific intermediate value. Of course, the adversary needs to repeat this procedure with multiple different basis points in order to correctly extract the information by correlating the hypothesis with the power traces. After he or she revealed the most significant bit of n , the adversary is able to determine the consecutive intermediate value and thus she or he can attack the next bit of n , cf. [Cor99].

The switching activity of the register transition of the intermediate value is

Algorithm 10 Determination of the activity A for attacking the Montgomery scalar multiplication on E

Require: A point \mathbb{P} on E , a positive integer $n=(n_{l-1}, \dots, n_0)_2$, the position m of the attacking key bit n_i with $l-2 \leq i < 0$, and the $l-i$ a priori known secret key bits

Ensure: The point $[n]\mathbb{P}$

```

1:  $\mathbb{P}_1 \leftarrow \mathbb{P}$  and  $\mathbb{P}_2 \leftarrow [2]\mathbb{P}$ 
2: for  $v = l-2$  down to 0 do
3:   if  $n_v = 0$  then
4:      $\mathbb{P}_1 \leftarrow [2]\mathbb{P}_1$  and  $\mathbb{P}_2 \leftarrow \mathbb{P}_1 \boxplus \mathbb{P}_2$ 
5:   else
6:      $\mathbb{P}_2 \leftarrow [2]\mathbb{P}_1$  and  $\mathbb{P}_1 \leftarrow \mathbb{P}_1 \boxplus \mathbb{P}_2$ 
7:      $\mathbb{P}_{temp} \leftarrow \mathbb{P}_1$ 
8:     /*Hypothesis for  $n_v=0$ */
9:      $\mathbb{P}_1 \leftarrow ([2]\mathbb{P}_1)$ 
10:     $A_{h_0(v)} \leftarrow \mathbb{P}_{1,x} \oplus \mathbb{P}_{temp,x}$ 
11:    /*Hypothesis for  $n_v=1$ */
12:     $\mathbb{P}_1 \leftarrow (\mathbb{P}_1 \boxplus \mathbb{P}_2)$ 
13:     $A_{h_1(v)} \leftarrow \mathbb{P}_{1,x} \oplus \mathbb{P}_{temp,x}$ 
14: return  $A_{h_0(i)}$  and  $A_{h_1(i)}$ 

```

needed in order to formulate the two required hypotheses. Algorithm 10 describes the procedure to determine the switching activity of the Montgomery ladder for a certain bit of the secret scalar n . The algorithm returns the XOR-results $A_{h_0(i)}$ and $A_{h_1(i)}$ of the previous and current register values storing the x-coordinate $\mathbb{P}_{1,x}$ of the current point \mathbb{P}_1 on the curve E , where $A_{h_0(i)}$ bases from the assumption the investigating $n_i=0$, as $A_{h_1(i)}$ refers to the hypothesis $n_i=1$. The point-doubling operation is denoted as $[2]$, while the addition operation is symbolized by \boxplus .

The obtained activities $A_{h_0(i)}$ and $A_{h_1(i)}$ can then be used with the Hamming distance model to gain the following leakage function:

$$\mathcal{L}_{\mathcal{F}_{t,ECC_ML,HD}}(n_i, A_{h_0(i)}, A_{h_1(i)}) \begin{cases} \text{HW}(A_{h_0(i)}) & \text{for } n_i = 0 \\ \text{HW}(A_{h_1(i)}) & \text{for } n_i = 1 \end{cases} \quad (6.1)$$

Please note that it is more reasonable to not use the complete bit representation of $A_{h_0(i)}$ or $A_{h_1(i)}$ for the Hamming weight operation, because most architectures do not store the intermediate value of the Montgomery ladder operation at one point in time. It is more likely that an ECC-coprocessor architecture is storing the intermediate value sequentially word-wise within the memory, causing a specific power consumption. In case the word length of the architecture is 32 bits and thus a 32 bit is chunk of $A_{h_0(i)}$ or $A_{h_1(i)}$ can be used for the leakage function.

It is also possible to consider the exploitable power consumption by a linear regression approach. The subspace $\mathcal{F}_{2,t;k}$ of $\mathcal{L}_{\mathcal{F}_{t,ECC,CSA^2}}(k_{(i)}, A_{h_{0(i)}}, A_{h_{0(i)}})$ can be used instead of the HD model in Eq. (6.1), where n_i is denoted as subkey $k_{(i)}$ in Eq. 6.2 to apply it to the stochastic approach.

$$g_{0,t;k}(k_{(i)}, A_{h_{0(i)}}, A_{h_{0(i)}}) = 1 \quad (6.2)$$

$$g_{1,t;k}(k_{(i)}, A_{h_{0(i)}}, A_{h_{0(i)}}) = \begin{cases} (A_{h_{0(i)}})_j - 16 & \text{for } k_{(i)} = 0 \\ (A_{h_{1(i)}})_j - 16 & \text{for } k_{(i)} = 1 \end{cases} \quad (6.3)$$

Properties of the mutating data path Due to the two threat scenarios the mutating data path should have the following properties:

- Executing the finite field multiplication operation on several different multiplier implementations
- Dynamically changing the amount of in parallel processing hardware units of finite field multipliers
- Changing the current representation of the intermediate values by different projective coordinates

The finite field multiplication is selected for the online allocation and the dynamic binding due to its gate complexity and thus, the higher power consumption compared to the addition and subtraction module. In addition, the finite field multiplication is a basic operation in both ECC-arithmetic operations point doubling and point-addition with a high utilization factor. This high utilization factor is perfectly exploitable in order to vary the degree of in parallel processing multiplication units. The different point-doubling and point-addition algorithms, which are used in the design, support up to three in parallel working finite field multipliers. Therefore, the type of multiplier implementation has a strong impact on the overall power consumption signature of the ECC-coprocessor.

The flexible binding method is applied on the coprocessor by randomizing the projective coordinate representation. The intermediate values, the current position on the basis point after some ECC-arithmetic operations, are represented in different coordinate systems. Instead of randomizing only one coordinate parameter representation like the randomized projective coordinates reported in [Cor99], the complete representation is changed.

6.2.1 Mutating Finite Field Multipliers

Algorithm 11 High-radix Montgomery multiplication from [MHAS08]

Require:

$$X = (x_{m-1}, \dots, x_1, x_0)_{2^r}, \text{ /* Operand } X^*/$$

$$Y = (y_{m-1}, \dots, y_1, y_0)_{2^r}, \text{ /* Operand } Y^*/$$

$$N = (n_{m-1}, \dots, n_1, n_0)_{2^r}, \text{ /* gcd(R,N)=1, with } X, Y < N < 2^k = R^* \text{ */}$$

$$w = -N^{-1} \bmod 2^r$$
Ensure: $Z = XY2^{-r \cdot m} \bmod N$

```

1:  $Z = 0$ 
2:  $v = 0$ 
3: for  $i = 0$  to  $m - 1$  do
4:    $(c_a, z_0) = z_0 + x_i y_0$ 
5:    $t_i = z_0 w \bmod 2^r$ 
6:    $(c_b, z_0) = z_0 + t_i n_0$ 
7:   for  $j = 0$  to  $m - 1$  do
8:      $(c_a, z_j) = z_j + x_i y_j + c_a$ 
9:      $(c_b, z_{j-1}) = z_j + t_i n_j + c_b$ 
10:     $(v_b, z_{m-1}) = c_a + c_b + v$ 
11:  if  $Z > N$  then
12:     $Z = Z - N$ 
13: return  $Z$ 

```

The design of the finite field multiplier of the ECC Mutate coprocessor is based on the high-radix Montgomery multiplier in order to process long numbers at a moderate resource consumption and acceptable execution duration. The online allocation concept is not applied on the multiplication algorithm, instead the core multiplication operation in the Montgomery high-radix algorithm, cf. Algorithm 11, is mutated in order to manipulate the power consumption signature. The implementation of the Montgomery multiplier is slightly different from Algorithm 11 by the mode of operation that it has a fixed runtime independent from the size of Z . At first, the design properties of the different core multiplier implementations are utilized in the high-radix Montgomery design, they are then analyzed in terms of resource consumption and critical path delay, followed by evaluating the different power consumption signatures of the designs, based on the methods introduced in Subsect. 3.2.2. The evaluation is conducted on real measurements derived by the measurement framework presented in Sect. 4.3.

6.2.1.1 Implementation of the Different Finite Field Multiplier Units

The core multiplier of the high-radix Montgomery is implemented by using ordinary CLB primitives of the Virtex 5 platform, thus it mainly utilizes LUTs and also some registers in order to reduce the critical data path. The architecture

of the core multiplier is segmented into three basic components in order to perform the multiplication based on a sequence of summations in an efficient way. The basic components are: a partial product generator (ppg), a partial product accumulator (ppa), and a carry propagation adder (cpa)². The concept of composing a finite field multiplier based on these components and their influence on the circuit design for an ASIC implementation from a CAD point of view as given in [MHAS08]. In the following the core multiplier composed by different implementations of the three basic components is investigated based on the results of a diploma thesis [Kus10] aimed at a partially reconfigurable Montgomery multiplier.

Core multiplier realization Based on the results of [Kus10] the ppa component has the biggest impact on the physical behavior in terms of the critical path delay and a power consumption of the core multiplier module. Thus, seven different core-multiplier implementations are investigated in terms of how different their specific power consumption signatures among each other are. The seven different ppa types are listed in Tab. 6.1. The designs differ in their tree structure and basic tree elements and thus affect the critical path delay, the resource consumption, as well as the power consumption. The design properties displayed in Tab. 6.1 base on an 8×8 -Bit multiplier. The selection of this word-size is justified due to the evaluation efforts in the following paragraph.

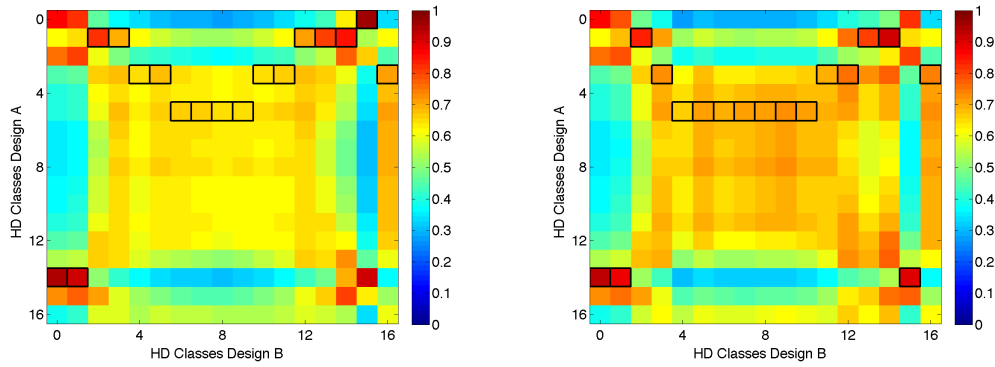
All the different multiplier implementations use the same ppg and cpa implementation, respectively. The partial product generator is a simple, straightforward implementation in order to generate the AND-wise combined partial products of the input bits of both operands. For the implementation of the cpa a Sklansky-adder implementation is used, because according to the diploma thesis it is the most resource efficient and fastest multiplier among the investigated different cpa stages on the Virtex 5 platform.

Evaluation of the different types The arithmetic circuit for evaluating the power consumption signature of the different ppa implementations consists of a simple ppg, one of the seven bit ppa implementations and a Sklansky-adder in suitable size as cpa component. The arithmetic is provided with two input registers at the operand inputs and at the result output in doubled length compared to the input register. The power consumption signature of generating the final result (the 8×8 multiplication result) is analyzed by the total correlation similar to the evaluation in chapter 5. The switching activity is separated into 17 Hamming distance classes according to the maximum bit-length of the result.

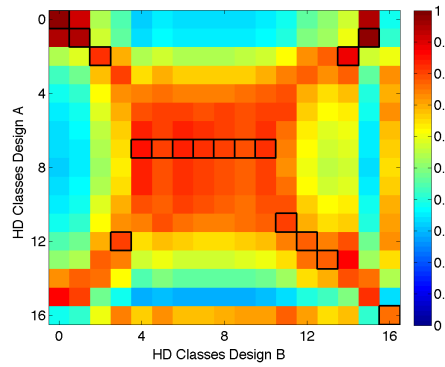
²The acronyms are written in lowercase letters to prevent a mixup between CPA (side-channel) and cpa (basic component multiplier)

Table 6.1: Resource consumption of the different SBox implementations

ppa	Reference	LUTs	Critical path delay
4:2 compressor	[OV93]	1893	7.7 ns
Wallace tree	[Wal64]	1875	8.3 ns
Overtured stairs	[MJ92]	2028	9.2 ns
Balanced tree	[ZM86]	1940	11.6 ns
Ripple carry	[Kus10]	1910	13.5 ns
Array 7:3	[Kus10]	1922	25.8 ns
Array	[Kus10]	1923	40.7 ns



(a) Overturned stairs(A) vs. Wallace-tree(B) (b) Overturned stairs(A) vs. balanced tree(B)



(c) Wallace-tree(A) vs. balanced tree(B)

Figure 6.3: Evaluation of the core multiplier components by the total correlation over the different Hamming distance classes

For each of the seven designs a set of 50,000 power traces has been captured under the condition to feed the same stimulus input to the different multiplier designs.

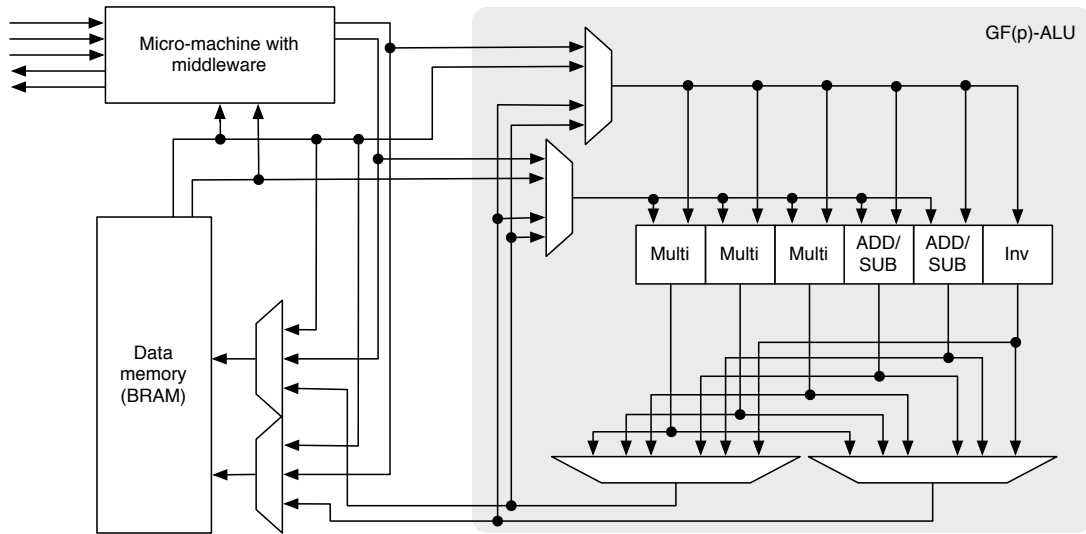


Figure 6.4: Interconnection for the data transfer

Out of the 24 design comparisons the three represented evaluation results in Fig. 6.3 have the most different power consumption signatures to each other. The ppa implementations based on the overturned stairs, the Wallace tree, and the balanced tree design have the most different power consumption signatures of their Hamming distance classes. The signatures of overturned stairs architecture to the Wallace tree architecture, as well as to the balanced tree architecture are significantly different, as none of the Hamming distance classes is mapped correspondently between those designs. Only the signature of the Wallace tree based implementation and the balanced tree implementation have similarities in the power consumption signature of seven Hamming distance classes (classes: 0, 1, 2, 11, 12, 13, and 16), cf. Fig. 6.3(c).

6.2.2 Management of Concurrent Multiplication Units

The management of the multiplication tasks is handled by the micro-machine and its virtualization schemes by distributing multiplication tasks to one of the multiple hardware multiplier units in the architecture. Due to the hierarchical architecture of the different arithmetic layers the coprocessor is highly modularized and able to process the different ECC-arithmetic operations on various configurations of basic finite field processing units. The flexibility of the micro-machine and virtualization unit is of high priority, so that for each point-doubling or point addition-operation a different data path configuration can be executed during the execution of the same superior scalar multiplication of the basis point.

Due to the virtualization scheme on the data path, the demultiplexing bus architecture and the shared BRAM data storage, the micro-machine in conjunction with the virtualizing schemes can configure the data path efficiently. For partially reconfiguring one of the hardware multiplier units the management unit has to distribute its workload on the remaining two multipliers. The VLIW instruction set allows the micro-machine and the virtualization unit to remap assigned multiplication tasks to another hardware unit. Thus the concept of dynamic binding in this architecture is similarly solved as in the application example of the AES. Also in this case the virtualization concept is the interface between the software part of the architecture and the hardware segment. The software controls the data flow by reconfiguring the data path structure of the ECC-coprocessor by the VLIW.

6.2.2.1 Virtualization Scheme

The central element of the virtualization scheme for the ECC Mutate is the VLIW instruction set. Table 6.2 depicts the architecture of the instruction set of the ECC-coprocessor. An opcode sequence in order to trigger one operation on the data path may have up to three different, sequential instruction types. These instruction types have a fixed bit length of 32 bits, but the information a bit position is containing changes based on the interpretation of the instruction type, cf. Tab. 6.2. The three instruction types can only appear in a strict sequence with special conditions of the previous instruction type. Therefore, the instruction types do not need a header for identification purposes.

The instruction Type 1 is first sent to set up the initialization phase of an executing operation. This instruction type contains the needed information to configure the $GF(p)$ -Alu component, cf. Fig. 6.4 and Tab. 6.2. In this phase the multiplexers in the $GF(p)$ -ALU as well as on the data memory are configured. The third least significant bit, bit 2, is an indication whether the second instruction type is following or maybe the third instruction type is the successor of the current instruction Type 1.

In case a memory access is indicated by the field *MA* the successor opcode is interpreted as an instruction Type 2 and sets the corresponding access information to the data memory. If a constant value access is indicated by the value "00" in the field *AI* or the field *BI* the last opcode command of the initialization phase is interpreted as instruction Type 3. After the initialization phase with one, two, or three opcode instructions the data transfer is started and the data is processed on the data path.

The internal virtualization scheme of the ECC-coprocessor differs from its implementation for the previous application on a symmetrical block cipher. The virtualization layer of the AES Mutate consists of micro-programs selected by

Table 6.2: VLIW structure of the ECC-coprocessor

Instruction Type 1

Bit	31:30	29:28	27:25	24:22	21:16	15:11	10:5	4:3	2	1	0
Function	AI	BI	RES1	RES2	loadA	loadB	outRES	AS	MA	RT	ST

Instruction Type 2

Bit	31:30	29:20	19	18	17:16	15:6	5	4	3:0
Function	DIA	ADDRA	ENA	WEA	DIB	ADDRB	ENB	WEB	not used

Instruction Type 3

Bit	31:28	27:24	23:0
Function	DIOPA	DIOPB	not used

Function	Description
AI /BI	Selects the input of port A/B of the $GF(p)$ -ALU.
RES1/RES2	Selects the output of a finite field arithmetic unit (FFAU) and provides it to the output 1/2 of the $GF(p)$ -ALU.
loadA/LoadB	Identifies the FFAU, if the current value on the internal busses are valid to process.
outRES	Informs the corresponding FFAU that the result is requested from the data transfer.
AS	Configures the corresponding subtractor-adder-unit to perform a subtraction or an addition.
MA	Request a memory access.
RT	Last command of the micro-program.
ST	Start signal for the FFAU, which was provided with new data.
DIA/DIB	Sets up the multiplexer at the data memory inputs.
ADDRA/ADDRB	Memory address of where to store in or to read from the data memory.
ENA/ENB	Enables the memory port A/B for data transfer.
WEA/WEB	Indicates if data is stored to the enabled port or data is read from the enabled port.
DIOPA/DIOPB	Addresses the constant value in the pre-calculated curve and algorithm specific constants.

a random number and a scrambling unit manipulating the incoming opcode sequences from the micro-programs. The virtualization layer of the ECC Mutate coprocessor only consist of a bigger set of micro-programs conducting the ECC-

Table 6.3: Supported point representations of the ECC Mutate coprocessor

Projective representation	Point representation
Affine	$\mathbb{P}(x,y)$
Standart projective without y	$\mathbb{P}(X/Z, -, Z)$ with $Z \neq 0$
Standart projective with y	$\mathbb{P}(X/Z, Y/Z, Z)$ with $Z \neq 0$
Jacobian with y	$\mathbb{P}(X/Z^2, Y/Z^3, Z)$ with $Z \neq 0$
Jacobian Chudnovsky with y	$\mathbb{P}(X/Z^2, Y/Z^3, Z, Z^2, Z^3)$ with $Z \neq 0$

arithmetic operations with a different set of hardware resources in various intermediate representations. For each utilization number of multipliers and possible value representations a micro-program is stored inside the BRAM.

6.2.3 Various Projective Coordinates

The flexible scheduling concept is applied differently in this application example compared to the previous example. The point representation is altered instead of merging the point-doubling and point-adding operations into various compositions of the scalar multiplication on the curve E . This mutation of the point representation is different compared to the countermeasure *randomized projective coordinates*, which is introduced in [Cor99]. The point representation does not randomize the Z component of the projective representation as in [Cor99], but the intermediate values are mapped to another point representation. Table 6.3 presents the different point representations the mutating coprocessor supports. Thereby, the internal calculations for the intermediate values are different in terms of their composition and the number of intermediate operational steps. The compositions of internal computation will of course influence the power consumption signature and will lead to an irregular pattern for each scalar multiplication. Usually, an adversary assumes that the irregularity of the power consumption signature is caused by differently processed values of the scalar multiplication $[n]\mathbb{P}$. Due to the usage of the Montgomery ladder as scalar multiplication the operation is balanced. The different patterns of the power consumption signature are independent to the scalar value and mislead an adversary to false conclusions about the secret value.

Additionally, the various internal sequences of calculating lead to randomization of the execution time and thus the begin and the end of each basic ECC-arithmetic operation is non-deterministic. This behavior of course increases the effort of mounting a successful DPA-like attack on the intermediate values of the Montgomery ladder, because of the shuffled execution of the operations in the time domain. The only disadvantage of utilizing various point representations

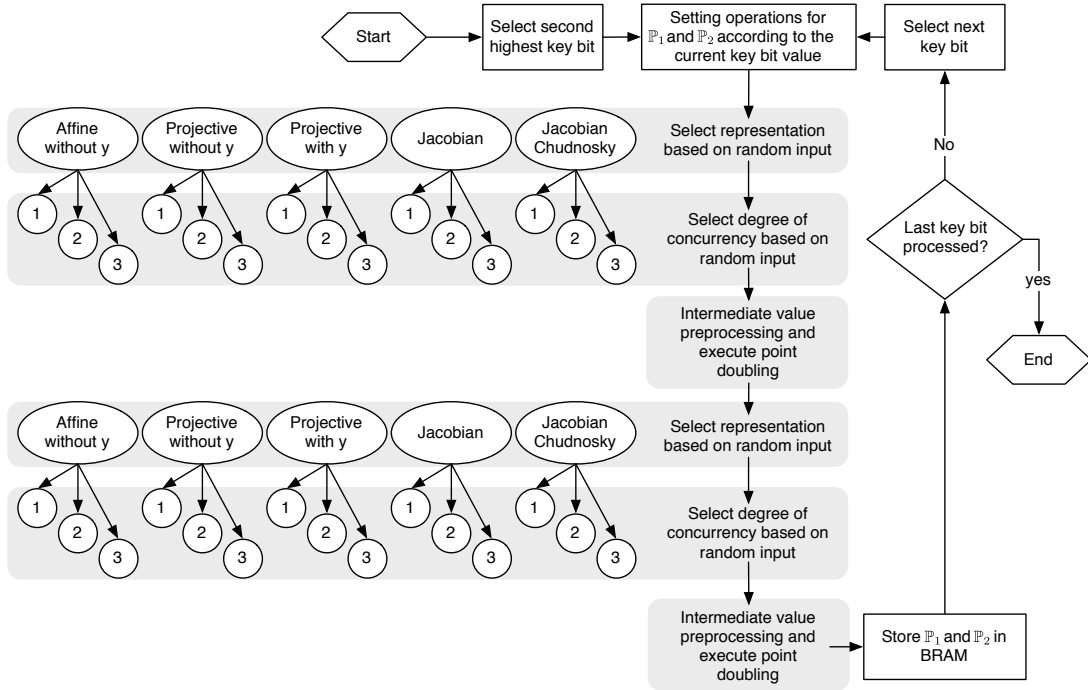


Figure 6.5: Composition of the scalar multiplication based on micro-programs

is the additional computational cost for the transformation from one point representation to the other. Also the complexity of the pre-computational steps of the transformation between the various representations depends on the kind of needed transformation to map from the current representation into the target representation. The expense differs for each case, for instance the transformation from the Jacobian coordinates representation to the affine point representation is more expensive than its counterpart from Jacobian to Jacobian Chudnosky.

All needed different transformation procedures between the supported point representations are stored in micro-programs. Also, the different sequences of procedures to perform a point-doubling or a point-addition with a certain number of multipliers are stored there. These micro-programs are managed and executed by the micro-machine with an embedded virtualization unit. A random number is used to select the projective representation and to set the number of utilized multipliers in order to perform the next ECC scalar multiplication. The random number thereby is used for an offset addressing scheme to call the required micro-program and to execute it on the micro-machine. The addressing scheme also allows an independent handling of the execution properties³ for both the point-

³Point representation and number of utilized multipliers

doubling and the point-addition operation. The monotonic execution sequence of the Montgomery ladder is exploited to integrate the random selection of the execution properties of the basic ECC-arithmetic operation. This is thereby achieved without exposing an exploitable key-dependent control flow of the scalar multiplication to an adversary. Figure 6.5 visualizes the control flow of the Montgomery ladder and the integration of the randomly selected micro-programs in order to execute the scalar multiplication in various ways.

6.2.4 Side-Channel Analysis of the Data Path

In this Subsection the side-channel evaluation results of the ECC-coprocessor design are discussed. The analysis on the ECC-coprocessor has been conducted by performing several scalar multiplications on the NIST 192 curve while varying the base point and keeping the scalar value static. The SASEBO-GII FPGA-Board is used as reference platform in order to capture the power traces of the implementations. The two previously discussed attack threats are investigated on the conducted experiments. At first, a SPA attack was investigated and then the attack on the intermediate values of the Montgomery ladder.

The evaluation of the mutating data path concept is conducted on two different mutating and one non-mutating ECC implementations. Two designs with a mutable data path have been implemented and evaluated, because of the reduced throughput of the ECC Mutate architecture with CLB primitives based multipliers, cf. Tab. 6.4. The first design is the ECC Mutate architecture using the DSP48E-slices to perform the basic multiplication within the high-radix Montgomery algorithm of the finite field multiplication. It is denoted as *ECC Mutate DSP* from now on. This design is completely based on the implementation presented in [Ema12]. Therefore, the mutation of the data path is only performed by the mutating methods dynamic binding and flexible scheduling. The second design, *ECC Mutate CLB* is an extension of the Mutate DSP design by using the three different previously analyzed core multipliers, which are implemented by using CLB primitives only. Thus, all three mutating methods, including the online allocation, are applicable on this implementation. This design has one partial reconfigurable multiplier and two fixed multiplier units using the ppa types overturned stairs and Wallace tree, cf. Subsect.6.2.1. Only one multiplier is set as partially reconfigurable at a time in order to process by operations at least on the other two multiplier units intended to hide the reconfiguration time.

For both designs the random numbers are provided by a Matlab script running in parallel. The same script is also used to control the partial reconfiguration of the finite field multiplier in order to manage the reconfiguration process and the random utilization of the in parallel processing multiplier units.

The unsecured reference version of the coprocessor-design stems from on the

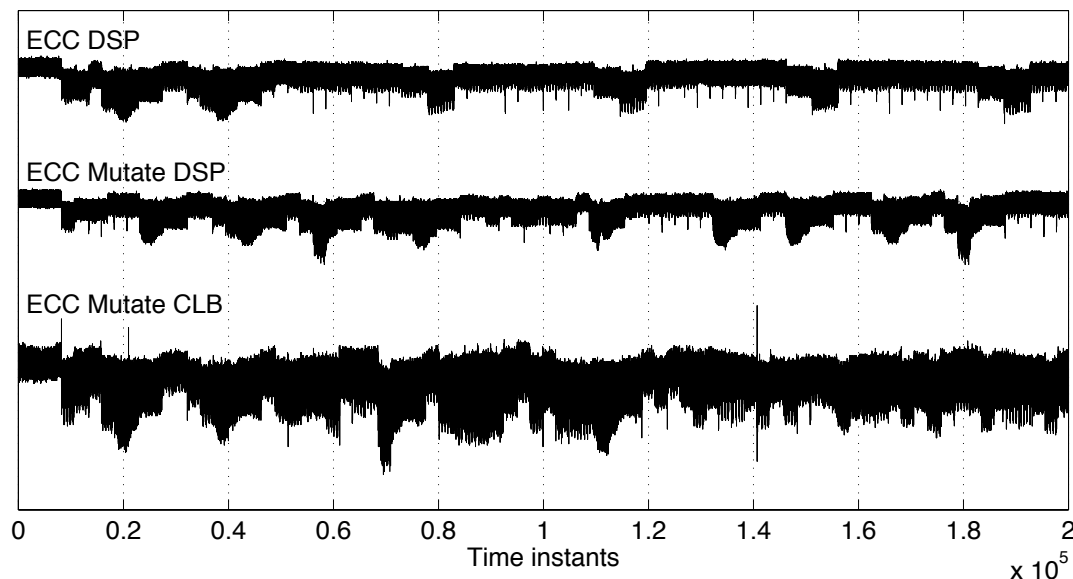


Figure 6.6: Captured traces for the SPA evaluation

Mutate DSP implementation and, thus, also uses the DSP48E primitives of the Virtex 5 platform (*ECC DSP*), but does not alter the data path. The unsecured coprocessor utilizes the Jacobian coordinates and three in parallel working finite field multiplier units to process the scalar point multiplication. In detail, the point-addition operation is performed with three finite field multipliers in parallel, while the point-doubling operation is performed with only one multiplier. The differences of the resource consumption and the required clock cycles to perform a scalar multiplication for different designs are listed in Tab. 6.4.

The analysis of the first threat is a direct visual interpretation of one captured power trace. A power trace from each of the three designs for this threat analysis is depicted in Fig. 6.6. The figure shows the processing of the first three key bit positions of the secret scalar. The first three values are taken as an example, as each bit causes the same process in the coprocessor. In all three cases the power consumption signature for either processing a zero bit or processing a none-zero bit value is indistinguishable. Thus, the Montgomery ladder protects the algorithm from leaking the control flow information, which may be exploited to reveal the secret bit sequence. The randomization of the two mutating coprocessor implementations increase the needed effort to clearly identify a regular pattern and thus to identify the point in time where the execution of a specific ECC-arithmetic is performed.

For the analysis of the second threat the exploitation of the intermediate

values of the Montgomery ladder operation is investigated. In specific the vulnerability of the storing x-component of the current intermediate value after processing the first unknown key bit is analyzed. An attack for this scenario needs to enable an adversary to freely select the basis point \mathbb{P} for each execution of the scalar multiplication $[n]\mathbb{P}$. Therefore, 20000 scalar multiplications with different basis points \mathbb{P} and the same scalar $[n]$ are performed on the three designs in order to capture enough power traces for the evaluation.

For all three analyzed cases no clear result leads to the right value of the first unknown secret bit of the used scalar value. Thus, it is not possible to make a clear statement on how much the mutating data path methods improve the security of the basic ECC DSP design. Only an approximation can be given based on the shuffling randomization by applying the mutating methods. Taking the randomization of the flexible scheduling (5 different representations), the randomization due to the dynamic binding (3 different degrees of concurrency), and the online allocation (3 different multiplier types) into consideration, 45 different states of the data path can occur within the time frame of capturing the power trace. So, the probability of capturing the power trace with the same data path configuration with a different basis point is about $\frac{1}{45}$, under the condition that the random number provided by the random number generator is equally distributed. The effort of attacking this hiding scheme should improve about the factor 45, if the adversary follows the strategy of gathering more traces and clustering them afterwards into 45 classes to attack a set of traces of one of these classes.

A possible reason for this unsatisfying result of the analysis may stem from the fact, that the SASEBO-GII board only provides measurement connections to monitor the power consumption of internal core-voltages. The main supply voltage of the BRAM primitives of the Virtex 5 is provided by the *auxiliary* supply voltage drain and thus, the monitoring of these bit flips by the internal core voltage is not precise. In fact, the information of the data-dependent power consumption of the bit flips inside the BRAM are propagated over the common ground plate in the FPGA, but also softened and disturbed by the internal decoupling capacitors inside the FPGA package. A more detailed investigation with more power traces and different key values may reveal clearer insight about the vulnerability to this attack scenario of both the secured and unsecured designs and to verify the above given hypothesis. However, in the limited time frame of this thesis it was not possible to conduct further experiments.

Table 6.4: Resource consumption of various ECC designs

Design	LUT	Flip flops	BRAM	DSP48E slices	Frequency [MHz]	$[n]\mathbb{P}$ [ms]	Platform
ECC P192 without countermeasures							
ECC DSP	12047	7719	9	9	166	18.75	Virtex 5
ECC P192 Mutate with countermeasures							
ECC Mutate DSP	12047	7719	9	9	165	15.23 - 40.24	Virtex 5
ECC Mutate CLB	14682	7747	9	6	72	32.64 - 86.23	Virtex 5
Fastest unsecured ECC design in literature on NIST curves							
P224 [Gün11]	1825	1892	11	26	486	0.45	Virtex 4

6.3 Discussion of the Evaluation Results

In the area of asymmetric cryptography most countermeasures against power analysis attacks focus on preventing SPA attacks, which exploit the control flow in order to reveal the secret value. Thus, a lot of countermeasures are focusing to balance the control flow or to hide the exploitable power consumption of the control flow. Countermeasures used for preventing DPA attacks on the intermediate values of the implemented algorithm are usually called *blinding* techniques. These blinding countermeasures are equivalent to the concept of masking countermeasures for securing block cipher algorithms and are more favored to be used for securing a design than hiding-based countermeasures.

The simple application of blinding-based countermeasures make them very attractive to secure the design, since most of them do not need any changes of the architecture or of the implemented operations. For instance, the finite field arithmetic does not need to be adapted for most of the blinding-based countermeasures. Therefore, the finite field operations are composed in order to get an ECC-arithmetic based operation only. Most of the blinding-based countermeasures exploit the homomorphic properties of the elliptic curve, for instance the scalar blinding technique, cf. [FV12]. The surveys [Cor99] and [FV12] provide a good overview of the commonly used or known countermeasures for elliptic curve cryptography. Especially [FV12] provides very useful results of a cost approximation analysis for the different countermeasures.

The dynamic mutating data path does not belong to the group of blinding countermeasures, this countermeasure is more related to the class of hiding methods. It aims at increasing the effort to identify points of interests in the captured power traces and tries to hinder the adversary to backtrace an exploitable relationship between the internal processed values and the power consumption signature. The cost of applying the mutating data path concept on an ECC $\text{GF}(p)$ scheme is both the increased resource consumption and delay of the discrete

implemented multiplier modulus compared to the specialized DSP48E primitives on the Virtex 5 platform. Therefore, the online allocation method is the most expensive method of the mutating data path for this application scheme. The other two mutating methods, dynamic binding and flexible scheduling, applied on the ECC-arithmetic layer follow the concept of the blinding-based countermeasure. The composition of the finite field basic operation is randomized for each execution of an ECC-arithmetic operation and thus it seems that the power consumption signatures are independent from the processed intermediate values.

The design introduced in [Gün11] is currently the fastest design in the literature for the NIST curves P-224 and P-256 implemented on the Virtex FPGA platform of Xilinx. It is speed and area optimized especially for only these two curves and uses a dedicated reduction algorithm. Thereby the design does not need a Montgomery algorithm to reduce the result of the finite field multiplication and thus saves several hundred cycles for the reduction of each modular multiplication leading to a significantly fast execution time of the curves P-224 and P-256.

On the other hand the ECC Mutate design, in general, can process every curve by utilizing the Montgomery higher-radix multiplication with integrated reduction, which leads to more clock cycles per multiplication. The speed loss and decreased throughput compared to the design reported in [Gün11] is explainable by the additional logic stage in order to realize a middleware layer for the virtualization process of the modular finite field arithmetic in order to complicate power-analysis attacks or to cope with malfunctional processing units. The article in [Gün11] does not report any countermeasures on the non-public design, thus only a theoretical approximation can be given of the decreased throughput in case of applying a blinding countermeasure on this fast design. For instance, exploiting the method of random key splitting, cf. [FV12], the throughput of the design in [Gün11] would be halved and the execution time would be doubled. A virtualization-based hiding method, like dynamic binding, is not possible because the design depends on only one modular multiplier unit. An approximation for other hiding-based countermeasures like random delay or clocking cannot be given without a deeper insight into the design, because the multi-clock based domain architecture of the ECC design in [Gün11].

The mutating data path countermeasure based ECC Mutate designs on the other hand can be combined with the previously mentioned blinding countermeasure concept in order to increase the needed effort during the trace acquisition. The combination of blinding-based countermeasures with the mutating data path concept is very suitable, because most of the blinding schemes, for instance like base point blinding, cf. [Cor99], or random key splitting, cf. [FV12], do not need to change the scalar multiplication coprocessor. Therefore, the blinding methods can be applied without any changes to the ECC Mutate coprocessor. The

application of both countermeasure methods, blinding and mutating data path, increases the resistance against power analysis attacks, because due to the mutating properties the profiling phase to build templates in a profiling-based power analysis attack is increased by the different states of the ECC data path. The nowadays very common technique of applying various types of countermeasures to a design in order to prevent side-channel exploitation is therefore applicable on the architectural level of the ECC algorithm in conjunction with the novel method of mutating data paths.

CHAPTER 7

Conclusion

Usually, countermeasures are applied either on the algorithmic level or on the cell level of a secured cryptographic algorithm implementation. The proposed hiding based countermeasure concept demonstrates that the effort of successfully attacking an implemented cryptographic algorithm via power analysis attacks may be increased by the mutating data path concept. From a conservative point of view the mutating data path concept combines several hiding methods, for instance shuffling and noise generation, in order to integrate them into a mutable data path architecture. The three different mutating methods of the proposed concept (online allocation, dynamic binding, and flexible scheduling) have been applied in this proof-of-concept investigation on two different algorithms after their conceptional discussion. The proposed design flow is used to span up a huge design exploration space compared to the initial design with many different design properties $\mathbf{dp}(\text{processing unit, concurrency, task})$, affecting the physical behavior of the design. The result of this design flow is a synthesized flexible, during the runtime mutable data path, which combines several hiding techniques. These techniques are embedded into the dynamical structure of the data path without compromising the computational correctness of the algorithm. The data path mutation is driven by a random selection, based on the output of a cryptographically secure random number generator. Thereby, even the designer can not determine the current active structure of the data path. The effort of exploiting side-channel leakage of the targeted implementation, based on profiling based methods, is increased under the assumption of a first order attack scenario.

Still the discussed concept of the mutating data path and the application examples illustrate the core concept of using a runtime dynamic data path, but also leave enough space for improvement or further investigation. Therefore, the following addressed themes are important to investigate in order to improve the concept or its application on different algorithm platforms.

Partial reconfiguration On the FPGA platforms, for instance the Virtex 5, the partial reconfiguration at first seems very suitable for the goal of partially changing the implementation of the data path by the concept of online allocation. The main challenge in applying the partial reconfiguration concept of Xilinx is the restriction to only be able to partially reconfigure a complete frame of the FPGA. The second disadvantage of this partial reconfiguration procedure is the

slow reconfiguration speed, especially if the reconfiguration is triggered from the outside of the device, as it is done for the application examples. An improvement to this circumstance would be to use the internal ICAP interface together with a fast hardware controller to conduct the partial reconfiguration. Smaller partial designs can be stored in several BRAM primitives on the FPGA, while bigger partial designs have to be stored externally. For the secure containment of these designs the bitstreams have to be encrypted and decrypted on the fly during the configuration, as shown in the concept presented in [SFH11].

A partial reconfiguration of smaller circuit parts of the data path is not efficient with the partial reconfiguration of Xilinx Inc., because due to the minimal size of one frame it is not resource efficient. An alternative subject of further investigation is to exploit the reconfigurable Virtex 5 primitive CFGLUT5 for the partial reconfiguration purpose of different implementation types for the on-line allocation. The challenging aspect of utilizing these primitives is that the designer has to take care by her- or himself of the dynamically changeable interconnection between the LUT.

Hiding concept The side-channel analysis on the application examples hardened by the concept of mutating data paths is done by a profiling based first order attack scenario. A second or higher order attack scenario is usually known for coping with masking or blinding based countermeasures in order to exploit the leakage of every processed part of the shared secret. A further interesting field to investigate may be to design a second order attack scheme or a combined side-channel analysis attack scheme tailored for the mutating data path hiding technique. Maybe it is possible to combine the information of the internal controller state, the power consumption signature of reconfiguration process, and the power consumption of the current active segments of the dynamic reconfigurable data path to successfully attack this kind of countermeasure.

Masking-based countermeasure schemes have been investigated intensively and developed over the last years in the side-channel community. One-well known threat on these countermeasure methods is struggling with data-dependent glitches based on the circuit design. Even so, theoretical concepts of preventing data-dependent glitches are not easy to implement and may still leak exploitable information as reported in [MM12]. This rises the question, whether the data-dependent and exploitable glitches of the masking scheme can be hidden by applying the mutating data path concept on top of a masking scheme in order to improve its resistance against higher-order attacks.

Improvement on the ECC application example As reported in the previous chapter one of the core concepts of the mutating data path method, the

online allocation, is sometimes affecting the designs speed, latency, or throughput negatively. In specific, performing the finite field multiplication in the ECC Mutate design on CLB primitives like LUTs and flip flops is affecting the performance a good deal worse than utilizing the especially for performance designed DSP multiplier primitives. One alternative to utilize different multiplier implementations for the online allocation is to introduce an additional layer between the finite field arithmetic and the ECC-arithmetic and to perform the online allocation on that layer. For instance, a residual number system (RNS) based design can be used to speed up the modular multiplication on the underlying finite field. The online allocation can then be performed on this layer by randomizing the different number bases of the RNS to change the power consumption signature of a higher-radix multiplication.

Basics to the Concept Chapter

A.1 Information Theoretic Background

- Definition of Joint Entropy: $H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2(p(x, y))$
- Definition of Entropy: $H(X) = -\sum_{x \in X} p(x) \log_2(p(x))$
- Definition of Conditional Entropy: $H(X|Y) = \sum_{x \in X} \sum_{y \in Y} p(x) \log_2(\frac{p(x)}{p(x, y)})$

A.2 CAD Algorithms

A.2.1 Left-Edge Algorithm

Algorithm 12 Left-edge algorithm from [Mic94]

Require: a set of intervals I describing a problem

```

1: Sort elements of I in a list of L in ascending order of  $l_i$ :
2:  $c=0$ 
3: while some interval has not been colored do
4:    $S=\emptyset$ 
5:    $r=0$ ; /* initialize coordinate of rightmost edge in  $S^*$  */
6:   while exists an element in L whose left edge coordinate is larger than r
     do
7:      $s$  = First element in the list L with  $l_s > r$ 
8:      $S = S \cup \{s\}$ 
9:      $r=r_r$ 
10:    Delete s from L
11:     $c=c+1$ 
12:    Label elements of S with color c

```

A.2.2 ASAP

Algorithm 13 *As Soon As Possible* scheduling from [Mic94]

Require: data flow graphs $DFG(V,E)$ of the implementing algorithm, d latency bound

```

1: schedule  $v_n$  by setting  $t_0^S=1$ 
2: repeat
3:   select vertex  $v_i$  whose successors are all scheduled
4:   schedule  $v_i$  by setting  $t_i^S = \min_{j:(v_j, v_i) \in E} t_j^S + d_i$ 
5: until  $v_0$  is scheduled
6: return  $t^L$ 

```

A.2.3 ALAP

Algorithm 14 *As Late As Possible* scheduling from [Mic94]

Require: data flow graphs $DFG(V,E)$ of the implementing algorithm, d latency bound

```

1: schedule  $v_n$  by setting  $t_n^L=d+1$ 
2: repeat
3:   select vertex  $v_i$  whose successors are all scheduled
4:   schedule  $v_i$  by setting  $t_i^L = \min_{j:(v_i, v_j) \in E} t_j^L - d_i$ 
5: until  $v_0$  is scheduled
6: return  $t^L$ 

```

Different Dimensional Subspaces

B.1 Higher Dimensional Subspaces of the AES Example Circuit

The higher u-dimensional subspace-based $\mathcal{L}_{\mathcal{M}, \text{CSA}^u}$ utilize the basis vector functions of the 9-dimensional subspace $\mathcal{L}_{\mathcal{M}, \text{CSA}^9}$. More precisely, the basis vector functions $g_{j,t;\mathbf{y}}(\mathbf{x}, \mathbf{y})$ of the $\mathcal{L}_{\mathcal{F}, \text{CSA}^9}(\mathbf{x}, \mathbf{y})$ of the $\mathcal{F}_{9,t;\mathbf{y}}$ without the weighting coefficients c_{g_j} (subtracted constant) is the basis to calculate the first 9 rows of matrix **A**. These basis vector functions are denoted by $g_{j,t;\mathbf{y}}^*(\mathbf{x}, \mathbf{y})$. Equation B.1 depicts $g_{j,t;k}^*(x, k)$ with $i=\{0, \dots, 9\}$ for the AES example circuit in Subsect. 4.1.2:

$$\begin{aligned} g_{0,t;k}^*(x, k) &= 1 \\ g_{j,t;k}^*(x, k) &= ((p \oplus k) \oplus \text{Sbox}(p \oplus k))_j \quad \text{for } j = 1, \dots, 8 \end{aligned} \quad (\text{B.1})$$

The other entities of the higher rows of the matrix **A** can also be composed out of the first nine columns of **A**. For instance the 37-dimensional subspace $\mathcal{F}_{37,t;\mathbf{y}}$ presented in [HSS12] constructs the basis vector functions 10 to 37 by multiplying two different rows element-wise. Thus, these elements of matrix **A** of the AES example circuit are equal to the result of the basis vectors:

$$\begin{aligned} g_{j,t;k}^*(x, k) &= ((p \oplus k) \oplus \text{Sbox}(p \oplus k))_{j_1} \cdot ((p \oplus k) \oplus \text{Sbox}(p \oplus k))_{j_2} \quad (\text{B.2}) \\ &= g_{j_1,t;k}^*(x, k) \cdot g_{j_2,t;k}^*(x, k) \\ &\quad \text{for } j = 9, \dots, 37 \text{ and } 1 \leq j_1 < j_2 \leq 1, \dots, 8 \end{aligned}$$

In order to fulfill the requirements $E(\mathcal{R}_t) = 0$ and $E(g_{j,t;\mathbf{y}}(\cdot))|_{\forall j} \neq 0$ for the subspace $\mathcal{F}_{37,t;\mathbf{y}}$ the weighting coefficients c_{g_j} have to be calculated. The coefficients c_{g_j} is the $E(g_{j,t;\mathbf{y}}^*(\mathbf{x}, \mathbf{y}))$, such as $E(g_{j,t;k}(x, k) - c_{g_j}) = 0 \mid \forall j \neq 0$. In case of the AES example circuit each c_{g_j} is calculated as follows:

$$c_{g_j} = \frac{1}{2^{16}} \cdot \sum_{k=0}^{255} \sum_{p=0}^{255} g_{j,t;k}^*(x, k) \quad \text{for } j = 1, \dots, 8 \quad (\text{B.3})$$

After c_{g_j} is calculated for each basis vector function, each of the j columns of matrix **A** are adjusted by the appropriate previous determined weight coefficient c_{g_j} . Higher dimensional subspaces, such as $\mathcal{F}_{163,t;\mathbf{y}}$ can be constructed in the same way.

B.2 Basis Vectors of a Lookup Table Functions

The basis vectors also exploits switching activities inside the basic components of the FPGA, which are contained in the Sbox circuit. The selected basic components for the subspace are three two-input LUTs and one five-input LUT. The columns above the ninth column are also composed of the entities of the first nine columns of matrix **A**. Therefore the same basis vector functions are used to calculate the first nine columns of matrix **A**, as shown in Eq. (B.1). The following Tab. B.1 depicts all basis vector functions $g_{j,t;k}^*(x, k)$ for a subspace $\mathcal{F}_{13,t,k}$. Each $g_{j,t;k}(x, k)$ of $\mathcal{L}_{\mathcal{F}_{\text{toy-AES}, \text{CSA}^{12}}}(k, p)$ can be determined by using Tab. B.1 Eq. (B.4).

$$g_{j,t;k}(x, k) = (g_{j,t;k}^*(x, k) - c_{g_j})|\forall j \quad (\text{B.4})$$

Table B.1: Construction of the $\mathcal{L}_{\mathcal{F}_{\text{toy-AES}, \text{CSA}^{12}}}(k, p)$

j	$g_{j,t;k}^*(x, k):$	$c_{g_j}:$
0	1	0
1	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_1$	0.5
2	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_2$	0.5
3	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_3$	0.5
4	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_4$	0.5
5	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_5$	0.5
6	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_6$	0.5
7	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_7$	0.5
8	$((p \oplus k) \oplus \text{Sbox}(p \oplus k))_8$	0.5
j	$F_{\text{lut}_{j-8}}((p \oplus k) \oplus \text{SBox}(p \oplus k))_j:$	$c_{g_j}:$
9	$g_{6,t;k}^*(x, k) \oplus g_{8,t;k}^*(x, k)$	0.7656
10	$g_{3,t;k}^*(x, k) \oplus g_{4,t;k}^*(x, k)$	0.7148
11	$g_{2,t;k}^*(x, k) \oplus g_{5,t;k}^*(x, k)$	0.7578
12	$(g_{6,t;k}^*(x, k) \wedge g_{8,t;k}^*(x, k) \wedge \neg g_{3,t;k}^*(x, k) \wedge \neg g_{4,t;k}^*(x, k)) \vee$ $(g_{6,t;k}^*(x, k) \wedge \neg g_{8,t;k}^*(x, k) \wedge g_{3,t;k}^*(x, k) \wedge \neg g_{4,t;k}^*(x, k)) \vee$ $(g_{6,t;k}^*(x, k) \wedge g_{8,t;k}^*(x, k) \wedge g_{3,t;k}^*(x, k) \wedge \neg g_{4,t;k}^*(x, k)) \vee$ $(g_{6,t;k}^*(x, k) \wedge \neg g_{8,t;k}^*(x, k) \wedge \neg g_{3,t;k}^*(x, k) \wedge g_{4,t;k}^*(x, k)) \vee$ $(\neg g_{6,t;k}^*(x, k) \wedge g_{8,t;k}^*(x, k) \wedge \neg g_{3,t;k}^*(x, k) \wedge g_{4,t;k}^*(x, k)) \vee$ $(\neg g_{6,t;k}^*(x, k) \wedge \neg g_{8,t;k}^*(x, k) \wedge g_{3,t;k}^*(x, k) \wedge g_{4,t;k}^*(x, k)) \vee$ $(g_{6,t;k}^*(x, k) \wedge g_{8,t;k}^*(x, k) \wedge g_{3,t;k}^*(x, k) \wedge g_{4,t;k}^*(x, k)) \vee$ $(\neg g_{6,t;k}^*(x, k) \wedge \neg g_{8,t;k}^*(x, k) \wedge \neg g_{3,t;k}^*(x, k) \wedge \neg g_{4,t;k}^*(x, k))$	0.0586

Bibliography

- [BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
- [Ber96] Philip A. Bernstein. Middleware: A Model for Distributed System Services. *Communication of the ACM*, 39(2):86–98, 1996.
- [BMM⁺03a] Luca Benini, Alberto Macii, Enrico Macii, Elvira Omerbegovic, Massimo Poncino, and Fabrizio Pro. A Novel Architecture for Power Maskable Arithmetic Units. In *GLSVLSI*, pages 136–140. ACM, 2003.
- [BMM⁺03b] Luca Benini, Alberto Macii, Enrico Macii, Elvira Omerbegovic, Fabrizio Pro, and Massimo Poncino. Energy-aware Design Techniques for Differential Power Analysis Protection. In *DAC*, pages 36–41. ACM, 2003.
- [BSCH11] Alexander Biedermann, Marc Stöttinger, Lijing Chen, and Sorin A. Huss. Secure Virtualization within a Multi-processor Soft-Core System-on-Chip Architecture. In Andreas Koch, Ram Krishnamurthy, John McAllister, Roger Woods, and Tarek A. El-Ghazawi, editors, *ARC*, volume 6578 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 2011.
- [Can05] D. Canright. A Very Compact Rijndael S-Box. Technical report, Naval Postgraduate School, 2005.
- [CB08] David Canright and Lejla Batina. A Very Compact "Perfectly Masked" S-Box for AES. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 5037 of *Lecture Notes in Computer Science*, pages 446–459. Springer, 2008.
- [CF05] Henri Cohen and Gerhard Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2005.
- [Cor99] Jean-Sébastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

- [CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
- [DES77] Data Encryption Standard. Technical report, U.S. Department of Commerce, 1977.
- [DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [DPA] <http://www.dpacontest.org/v2/index.php>.
- [DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate Side Channel Attacks and Leakage Modeling. *Journal of Cryptographic Engineering*, 1:123–144, 2011.
- [EG10] M. Abdelaziz Elaabid and Sylvain Guilley. Practical Improvements of Profiled Side-Channel Attacks on a Hardware Crypto-Accelerator. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 243–260. Springer, 2010.
- [EJM⁺02] Markus Ernst, Michael Jung, Felix Madlener, Sorin A. Huss, and R. Blümel. A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over $\text{GF}(2^n)$. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 381–399. Springer, 2002.
- [Ema11] Morteza Emamgholi. Sichere Hardware für Car-2-X Kommunikation, 2011.
- [Ema12] Morteza Emamgholi. Krypto-Prozessor über elliptische in endlichen Körpern, 2012.
- [FG05] Wieland Fischer and Berndt M. Gammel. Masking at Gate Level in the Presence of Glitches. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2005.
- [FMPR10] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine Masking against Higher-Order Side Channel Analysis. In Alex Biryukov, Guang Gong, and Douglas R. Stinson,

- son, editors, *SAC*, volume 6544 of *Lecture Notes in Computer Science*, pages 262–280. Springer, 2010.
- [FPG] <http://www.1-core.com/library/digital/fpga-architecture/fpga-architecture.pdf>.
- [FV12] Junfeng Fan and Ingrid Verbauwhede. An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost. In David Naccache, editor, *Cryptography and Security*, volume 6805 of *Lecture Notes in Computer Science*, pages 265–282. Springer, 2012.
- [GAGS09] Daniel D. Gajski, Samar Abdi, Andreas Gerstlauer, and Gunar Schirner. *Embedded System Design: Modeling, Synthesis and Verification*. Springer Publishing Company, Incorporated, 1st edition, 2009.
- [GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
- [GFD09] Patrick Gallagher, Deputy Director Foreword, and Cita Furlani Director. FIPS PUB 186-3 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS), 2009.
- [GHP04] Sylvain Guilley, Philippe Hoogvorst, and Renaud Pacalet. Differential Power Analysis Model and Some Results. In Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and Anas Abou El Kalam, editors, *CARDIS*, pages 127–142. Kluwer, 2004.
- [GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. Stochastic Methods. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2006.
- [GM11] Tim Güneysu and Amir Moradi. Generic Side-Channel Countermeasures for Reconfigurable Devices. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2011.
- [GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David

- Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
- [GMS⁺11] S. Guilley, H. Maghrebi, Y. Souissi, L. Sauvage, and J.L. Danger. Quantifying the Quality of Side-Channel Acquisition. In *COSADE*, pages 16–28, 2011.
- [GrM04] Jovan Dj. Golic and r. Menicocci. Universal Masking on Logic Gate Level. *IEEE Electronic Letters*, 40(9):526–528, April 2004.
- [GT02] Jovan Dj. Golic and Christophe Tymen. Multiplicative Masking and Power Analysis of AES. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.
- [Gün11] Tim Güneysu. Utilizing Hard Cores of Modern FPGA Devices for High-Performance Cryptography. *Journal of Cryptographic Engineering*, 1(1):37–55, 2011.
- [HdlTR12] Wei He, Eduardo de la Torre, and Teresa Riesgo. An Interleaved EPE-Immune PA-DPL Structure for Resisting Concentrated EM Side Channel Attacks on FPGA Implementation. In Werner Schindler and Sorin A. Huss, editors, *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2012.
- [HF12] Anh-Tuan Hoang and Takeshi Fujino. Intra-Masking Dual-Rail Memory on LUT Implementation for Tamper-Resistant AES on FPGA. In Katherine Compton and Brad L. Hutchings, editors, *FPGA*, pages 1–10. ACM, 2012.
- [HKSS11] Annelie Heuser, Michael Kasper, Werner Schindler, and Marc Stöttinger. How a Symmetry Metric Assists Side-Channel Evaluation - A Novel Model Verification Method for Power Analysis. In *DSD*, pages 674–681. IEEE, 2011.
- [HMOV03] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2003.
- [HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2006.

- [HSS12] Annelie Heuser, Werner Schindler, and Marc Stöttinger. Revealing Side-Channel Issues of Complex Circuits by Enhanced Leakage Models. In Wolfgang Rosenstiel and Lothar Thiele, editors, *DATE*, pages 1179–1184. IEEE, 2012.
- [JB07] R.C. Jaeger and T.N. Blalock. *Microelectronic Circuit Design*. McGraw-Hill series in electrical and computer engineering. McGraw-Hill Higher Education, 2007.
- [Jef46] Harold Jeffreys. An Invariant Form for the Prior Probability in Estimation Problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):pp. 453–461, 1946.
- [JSG⁺12] Bernhard Jungk, Marc Stöttinger, Jan Gampe, Steffen Reith, and Sorin A. Huss. Side-Channel Resistant AES Architecture Utilizing Randomized Composite Field Representations. In *FPT*. IEEE, 2012. *Accepted, to be published in December*.
- [JT01] Marc Joye and Christophe Tymen. Protections against Differential Analysis for Elliptic Curve Cryptography. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 377–390. Springer, 2001.
- [KB07] Boris Köpf and David A. Basin. An Information-Theoretic Model for Adaptive Side-Channel Attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *CCS*, pages 286–296. ACM, 2007.
- [KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
- [KL51] Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *Annual Mathematical Statistics*, 22(1):79–86, 1951.
- [Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
- [Koc08] Cetin Kaya Koc. *Cryptographic Engineering*. Springer Publishing Company, Incorporated, 1st edition, 2008.

- [KSS10] Michael Kasper, Werner Schindler, and Marc Stöttinger. A Stochastic Method for Security Evaluation of Cryptographic FPGA Implementations. In Jinian Bian, Qiang Zhou, Peter Athanas, Yajun Ha, and Kang Zhao, editors, *FPT*, pages 146–153. IEEE, 2010.
- [Kus10] Dennis Kusidlo. Entwurf eines Partiell Rekonfigurierbaren Montgomery Multiplizierers, 2010.
- [LBHO10] Yingxi Lu, Keanhong Boey, Philip Hodggers, and Máire O’Neill. Lightweight DPA Resistant Solution on FPGA to Counteract Power Models. In Jinian Bian, Qiang Zhou, Peter Athanas, Yajun Ha, and Kang Zhao, editors, *FPT*, pages 178–183. IEEE, 2010.
- [LLV] <http://11vm.org/docs/>.
- [LMT⁺09] Victor Lomné, Philippe Maurine, Lionel Torres, Michel Robert, Rafael Soares, and Ney Calazans. Evaluation on FPGA of Triple Rail Logic Robustness against DPA and DEMA. In *DATE*, pages 634–639. IEEE, 2009.
- [LOM10] Yingxi Lu, Máire O’Neill, and John V. McCanny. Evaluation of Random Delay Insertion against DPA on FPGAs. *TRETS*, 4(1):11, 2010.
- [LRP07] Kerstin Lemke-Rust and Christof Paar. Analyzing Side Channel Leakage of Masked Implementations with Stochastic Methods. In Joachim Biskup and Javier Lopez, editors, *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 454–468. Springer, 2007.
- [Man02] Stefan Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC*, volume 2587 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2002.
- [MGV08] Nele Mentens, Benedikt Gierlichs, and Ingrid Verbauwhede. Power and Fault Analysis Resistance in Hardware through Dynamic Reconfiguration. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 346–362. Springer, 2008.
- [MHAS08] Atsushi Miyamoto, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. Systematic Design of HighRadix Montgomery Multipliers for RSA Processors. In *ICCD*, pages 416–421. IEEE, 2008.

- [Mic94] Giovanni De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1st edition, 1994.
- [MJ92] Zhi-Jian Mou and Francis Jutand. Overturned-Stairs Adder Trees and Multiplier Design. *Computers, IEEE Transactions on*, 41(8):940–948, aug 1992.
- [MM12] Amir Moradi and Oliver Mischke. Glitches and static power hand in hand. *IACR Cryptology ePrint Archive*, 2012:472, 2012.
- [MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010.
- [MOS09] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for All - All for One: Unifying Standard DPA Attacks. *IACR Cryptology ePrint Archive*, 2009:449, 2009.
- [MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
- [MPL⁺11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.
- [MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
- [MPO07] Stefan Mangard, Thomas Popp, and Maria Elisabeth Oswald. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.
- [MR04] Silvio Micali and Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

- [MS02] Sumio Morioka and Akashi Satoh. An Optimized S-Box Circuit Architecture for Low Power AES Design. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2002.
- [MSH09] Felix Madlener, Marc Stöttinger, and Sorin Alexander Huss. Novel Hardening Techniques against Differential Power Analysis for Multiplication in $GF(2^n)$. In *FPT*. IEEE, 2009.
- [MSSS11] H. Gregor Molter, Marc Stoettinger, Abdulhadi Shoufan, and Falko Strenzke. A Simple Power Analysis Attack on a McEliece Cryptoprocessor. *Journal of Cryptographic Engineering*, 1:29–36, January 2011.
- [NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.
- [NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *Journal of Cryptology*, 24(2):292–321, 2011.
- [OO07] Levent Ordu and Siddika Berna Ors. Power Analysis Resistant Hardware Implementations of AES. In *ICECS*, pages 1408–1411. IEEE, dec. 2007.
- [OV93] Vojin G. Oklobdzija and David Villeger. Multiplier Design Utilizing Improved Column Compression Tree and Optimized Final Adder in CMOS Technology. In *VLSI Technology, Systems, and Applications*, pages 209–212. IEEE, 1993.
- [Pag02] Dan Page. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. *IACR Cryptology ePrint Archive*, 2002:169, 2002.
- [PKZM07] Thomas Popp, Mario Kirschbaum, Thomas Zefferer, and Stefan Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2007.
- [PM05] Thomas Popp and Stefan Mangard. Masked Dual-Rail Precharge Logic: DPA-Resistance Without Routing Constraints. In

- Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.
- [PP10] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.
- [PSQ07] Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. *Integration*, 40(1):52–60, 2007.
- [QS01] Jean-Jacques Quisquater and David Samyde. Electro-Magnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In Isabelle Attali and Thomas P. Jensen, editors, *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
- [RBNB04] Rahul M. Rao, Richard B. Brown, Kevin Nowka, and Jeffrey B. Burns. Analysis and mitigation of CMOS gate leakage. In *Fifth Annual Austin Center for Advanced Studies Conference*, pages 7–11, 2004.
- [RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communication of the ACM*, 21(2):120–126, 1978.
- [RYS11] Francesco Regazzoni, Wang Yi, and Francois-Xavier Standaert. FPGA Implementations of the AES Masked Against Power Analysis Attacks. In *COSADE*, pages 55–66, 2011.
- [SAS] <http://www.rcis.aist.go.jp/special/SASEBO/SASEBO-GII-en.html>.
- [SBH10] Marc Stöttinger, Alexander Biedermann, and Sorin Alexander Huss. Virtualization within a Parallel Array of Homogeneous Processing Units. In Phaophak Sirisuk, Fearghal Morgan, Tarek A. El-Ghazawi, and Hideharu Amano, editors, *ARC*, volume 5992 of *Lecture Notes in Computer Science*, pages 17–28. Springer, 2010.
- [Sch05] Werner Schindler. On the Optimization of Side-Channel Attacks by Advanced Stochastic Methods. In Serge Vaudenay, editor, *PKC*, volume 3386 of *Lecture Notes in Computer Science*, pages 85–103. Springer, 2005.

- [Sch08] Werner Schindler. Advanced Stochastic Methods in Side Channel Analysis on Block Ciphers in the Presence of Masking. *Journal of Mathematical Cryptology*, 2:291–310, 2008.
- [SE08] Frederic Stumpf and Claudia Eckert. Enhancing Trusted Platform Modules with Hardware-Based Virtualization Techniques. In *SECURWARE*, pages 1–9. IEEE, 2008.
- [SFH11] Marc Stöttinger, Thomas Feller, and Sorin A. Huss. A Side-Channel hardened IP-Protection Scheme for FPGA-based Platforms. In Russell Tessier, editor, *FPT*. IEEE, 2011.
- [SHAS09] T. Sugawara, N. Homma, T. Aoki, and A. Satoh. Differential Power Analysis of AES ASIC Implementations with Various S-box Circuits. In *ECCTD*. IEEE, 2009.
- [SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.
- [SMH09] Marc Stöttinger, Felix Madlener, and Sorin A. Huss. Procedures for Securing ECC Implementations against Differential Power Analysis Using Reconfigurable Architectures. In Marco Platzner, Jürgen Teich, and Norbert Wehn, editors, *Dynamically Reconfigurable Systems - Architectures, Design Methods and Applications*, pages 305–321. Springer, 2009.
- [SMTM01] Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 239–254. Springer, 2001.
- [SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.
- [SSI04] Daisuke Suzuki, Minoru Saeki, and Tetsuya Ichikawa. Random Switching Logic: A Countermeasure against DPA based on Transition Probability. *IACR Cryptology ePrint Archive*, 2004:346, 2004.
- [SSMS09] Abdulhadi Shoufan, Falko Strenzke, H. Gregor Molter, and Marc Stöttinger. A Timing Attack Against Patterson Algorithm in the

- McEliece PKC. In Donghoon Lee and Seokhie Hong, editors, *ICISC*, volume 5984 of *Lecture Notes in Computer Science*, pages 161–175. Springer, 2009.
- [SVCO⁺10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The World is Not Enough: Another Look on Second-Order DPA. *IACR Cryptology ePrint Archive*, 2010:180, 2010.
- [SVKH10] Shaunak Shah, Rajesh Velegalati, Jens-Peter Kaps, and David Hwang. Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs. In Viktor K. Prasanna, Jürgen Becker, and René Cumplido, editors, *ReConFig*, pages 274–279. IEEE, 2010.
- [SW12] Sergei Skorobogatov and Christopher Woods. In the blink of an eye: There goes your aes key. *IACR Cryptology ePrint Archive*, 2012:296, 2012.
- [SWM⁺10] Abdulhadi Shoufan, Thorsten Wink, H. Gregor Molter, Sorin A. Huss, and Eike Kohnert. A Novel Cryptoprocessor Architecture for the McEliece Public-key Cryptosystem. *IEEE Transactions on Computers*, 99(PrePrints), 2010.
- [Tha09] Christian Thaler. Entwurf eines KOM-Interfaces für DPA-Messungen, July 2009.
- [THH12] Qizhi Tian, Annelie Heuser, and Sorin A. Huss. A Novel Analysis Method for Assessing the Side-Channel Resistance of Cryptosystems. In *IIHMS*, pages 126–129. IEEE, 2012.
- [TKL04] Elena Trichina, Tymur Korkishko, and Kyung-Hee Lee. Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *AES Conference*, volume 3373 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 2004.
- [TS01] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall PTR, 2001.
- [TSG02] Elena Trichina, Domenico De Seta, and Lucia Germani. Simplified Adaptive Multiplicative Masking for AES. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 187–197. Springer, 2002.

- [TV04] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *DATE*, pages 246–251. IEEE, 2004.
- [ug1] Virtex-5 fpga user guide. Technical report, Xilinx. http://www.xilinx.com/support/documentation/user_guides/ug190.pdf.
- [UG612] Virtex-5 Libraries Guide for HDL Design. Technical report, Xilinx, 2012. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/virtex5_hdl.pdf.
- [Vee84] H.J.M. Veendrick. Short-Circuit Dissipation of Static CMOS circuitry and its Impact on the Design of Buffer Circuits. *Solid-State Circuits, IEEE Journal of*, 19(4):468 – 473, 1984.
- [VK11] Rajesh Velegalati and Jens-Peter Kaps. Improving Security of SDDL Designs through Interleaved Placement on Xilinx FPGAs. In *FPL*, pages 506–511. IEEE, 2011.
- [Wal64] Chris. S. Wallace. A Suggestion for a Fast Multiplier. *Electronic Computers, IEEE Transactions on*, EC-13(1):14–17, 1964.
- [WE93] Neil H.E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI design: a systems perspective*. VLSI systems series. Addison-Wesley Pub. Co., 1993.
- [WO11a] Carolyn Whitnall and Elisabeth Oswald. A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2011.
- [WO11b] Carolyn Whitnall and Elisabeth Oswald. A Fair Evaluation Framework for Comparing Side-Channel Distinguishers. *Journal of Cryptographic Engineering*, 1(2):145–160, 2011.
- [WOS12] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The Myth of Generic DPA...and the Magic of Learning. *IACR Cryptology ePrint Archive*, 2012:256, 2012.
- [Yao03] Yiyu Yao. *Entropy Measures, Maximum Entropy and Emerging Applications*, chapter Information-theoretic measures for knowledge discovery and data mining, pages 115–136. Springer, 2003.
- [Yeh03] C.H. Yeh. The robust middleware approach for transparent and systematic fault tolerance in parallel and distributed systems. In *ICPP*, pages 61–68. IEEE, 2003.

-
- [YS07] Pengyuan Yu and Patrick Schaumont. Secure FPGA Circuits Using Controlled Placement and Routing. In Soonhoi Ha, Kiyong Choi, Nikil D. Dutt, and Jürgen Teich, editors, *CODES+ISSS*, pages 45–50. ACM, 2007.
- [ZM86] Dan Zuras and William H. McAllister. Balanced Delay Trees And Combinatorial Division in VLSI. *Solid-State Circuits, IEEE Journal of*, 21(5):814 – 819, oct 1986.

Own Publications

- [1] Bernhard Jungk, Marc Stöttinger, Jan Gampe, Steffen Reith, and Sorin A. Huss. Side-Channel Resistant AES Architecture Utilizing Randomized Composite Field Representations. In *FPT*. IEEE, 2012. *Accepted, to be published in December*.
- [2] Qizhi Tian, Abdulhadi Shoufan, Marc Stöttinger, and Sorin A. Huss. Power Trace Alignment for Cryptosystems featuring Random Frequency Countermeasures. In *ICDIPC*, 2012.
- [3] Michael Zohner, Marc Stöttinger, Sorin A. Huss, and Oliver Stein. An Adaptable, Modular, and Autonomous Side-Channel Vulnerability Evaluator. In *HOST*, 2012.
- [4] Michael Zohner, Michael Kasper, and Marc Stöttinger. Butterfly-Attack on Skein’s Modular Addition. In Werner Schindler and Sorin A. Huss, editors, *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 215–230. Springer, 2012.
- [5] Annelie Heuser, Werner Schindler, and Marc Stöttinger. Revealing Side-Channel Issues of Complex Circuits by Enhanced Leakage Models. In Wolfgang Rosenstiel and Lothar Thiele, editors, *DATE*, pages 1179–1184. IEEE, 2012.
- [6] Marc Stöttinger, H. Gregor Molter, and Sorin A. Huss. Acceleration of Hypotheses Matrix Generation for DPA Attacks. In Wolfgang Rosenstiel and Lothar Thiele, editors, *DATE*. IEEE, 2012.
- [7] Michael Zohner, Michael Kasper, Marc Stöttinger, and Sorin A. Huss. Side Channel Analysis of the SHA-3 Finalists. In Wolfgang Rosenstiel and Lothar Thiele, editors, *DATE*, pages 1012–1017. IEEE, 2012.
- [8] Annelie Heuser, Michael Kasper, Werner Schindler, and Marc Stöttinger. A Difference Method for Side-Channel Analysis Exploiting High-Dimensional Leakage Models. In *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 365–382. Springer, 2012.
- [9] Marc Stöttinger, Thomas Feller, and Sorin A. Huss. A Side-Channel hardened IP-Protection Scheme for FPGA-based Platforms. In Russell Tessier, editor, *FPT*. IEEE, 2011.

- [10] Michael Zohner, Michael Kasper, and Marc Stöttinger. Side Channel Evaluation of the SHA-3 Candidates. In *TrustED*, September 2011.
- [11] Annelie Heuser, Michael Kasper, Werner Schindler, and Marc Stöttinger. How a Symmetry Metric Assists Side-Channel Evaluation - A Novel Model Verification Method for Power Analysis. In *DSD*, pages 674–681. IEEE, 2011.
- [12] Alexander Biedermann, Marc Stöttinger, Lijing Chen, and Sorin A. Huss. Secure Virtualization within a Multi-processor Soft-Core System-on-Chip Architecture. In Andreas Koch, Ram Krishnamurthy, John McAllister, Roger Woods, and Tarek A. El-Ghazawi, editors, *ARC*, volume 6578 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 2011.
- [13] H. Gregor Molter, Marc Stöttinger, Abdulhadi Shoufan, and Falko Strenzke. A Simple Power Analysis Attack on a McEliece Cryptoprocessor. *Journal of Cryptographic Engineering*, 1:29–36, January 2011.
- [14] Marc Stöttinger, Sorin A. Huss, Sascha Mühlbach, and Andreas Koch. Side-Channel Resistance Evaluation of a Neural Network Based Lightweight Cryptography Scheme. In *EUC*, pages 603–608. IEEE, 2010.
- [15] Michael Kasper, Werner Schindler, and Marc Stöttinger. A Stochastic Method for Security Evaluation of Cryptographic FPGA Implementations. In Jinian Bian, Qiang Zhou, Peter Athanas, Yajun Ha, and Kang Zhao, editors, *FPT*, pages 146–153. IEEE, 2010.
- [16] Marc Stöttinger, Sunil Malipatlolla, and Qizhi Tian. Survey of Methods to Improve Side-Channel Resistance on Partial Reconfigurable Platforms. In Alexander Biedermann and Gregor Molter, editors, *Design Methodologies for Secure Embedded Systems*. Springer, 2010.
- [17] Marc Stöttinger, Alexander Biedermann, and Sorin Alexander Huss. Virtualization within a Parallel Array of Homogeneous Processing Units. In Phaophak Sirisuk, Fearghal Morgan, Tarek A. El-Ghazawi, and Hideharu Amano, editors, *ARC*, volume 5992 of *Lecture Notes in Computer Science*, pages 17–28. Springer, 2010.
- [18] Marc Stöttinger, Felix Madlener, and Sorin A. Huss. Procedures for Securing ECC Implementations against Differential Power Analysis Using Reconfigurable Architectures. In Marco Platzner, Jürgen Teich, and Norbert Wehn, editors, *Dynamically Reconfigurable Systems - Architectures, Design Methods and Applications*, pages 305–321. Springer, 2009.

-
- [19] Felix Madlener, Marc Stöttinger, and Sorin Alexander Huss. Novel Hardening Techniques against Differential Power Analysis for Multiplication in $\text{GF}(2^n)$. In *FPT*. IEEE, 2009.
 - [20] Abdulhadi Shoufan, Falko Strenzke, H. Gregor Molter, and Marc Stöttinger. A Timing Attack Against Patterson Algorithm in the McEliece PKC. In Donghoon Lee and Seokhie Hong, editors, *ICISC*, volume 5984 of *Lecture Notes in Computer Science*, pages 161–175. Springer, 2009.

List of supervised student thesis and student projects

1. Kai Rohde, *Automated Integration of Power Analysis Attack Countermeasures for Hardware Designs*, master thesis, October 2012
2. Omid Pahlevan Sharif, *Implementation of a GrøstlAES Coprocessor with Side-Channel Analysis*, bachelor thesis, August 2012
3. Christian Thaler, *Evaluation einer IMDPL DPA-Gegenmaßnahme für FPGA Plattformen*, student project, August 2012
4. Morteza Emamgholi, *Krypto-Prozessor über elliptische in endlichen Körpern*, student project, Juni 2012
5. Joannes Klaus, *Side-Channel Analysis of a MicroBlaze Based AES Implementation*, bachelor thesis, August 2011
6. Uwe Breidenbach, Oliver Dietz and Christopher Huth, *Evaluation of a Masked AES implementation*, semester lab, April 2011
7. Morteza Emamgholi and Daniel Münch, *Secure Hardware for Car-2-X Communication*, bachelor thesis, March 2011
8. Jonas Schönichen, *Advanced Analysis Methods for DPA*, bachelor thesis, Januar 2011
9. Dennis Kusidlo, *Design and Implementation of a Partial Reconfigurable Montgomery multiplier*, diploma thesis, November 2010
10. Octavio Gamero, *Design and Implementation of an Evaluation Platform for Side-Channel Analysis*, diploma thesis, October 2010
11. Christian Rückriegel, *Implementation of a Generic ECC Controller*, bachelor thesis, August 2010
12. Steven Arzt, Daniel Münch, Gregor Rynkowski and Alexander Spitzl, *Accelerated Calculation of Hypotheses for Side-Channel Attacks*, bachelor semester lab, April 2010
13. Christian Thaler, *Implementation of a KOM-Interface for DPA Measurements*, bachelor thesis, July 2009